AD-A100 799 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 6/5
ANALYSIS AND PERFORMANCE EVALUATION OF ELECTROCARDIOGRAM DATA C--ETC(U)
DEC 80 M D TOWNSEND

UNCLASSIFIED AFIT/GE/EE/80D-46 NL

1 OF 3

AD A
100799

AFIT/GE/EE/80D-46

Accession For
NTIS GRA&I X
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Avail and/or
Dist Special
A

ANALYSIS AND PERFORMANCE EVALUATION

OF ELECTROCARDIOGRAM

DATA COMPRESSION TECHNIQUES

THESIS

AFIT/GE/EE/80D-46 Melvin D. Townsend
Captain USAF

Approved for public release; distribution unlimited

AFIT/GE/EE/80D-46

ANALYSIS AND PERFORMANCE EVALUATION OF ELECTROCARDIOGRAM DATA COMPRESSION TECHNIQUES

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

Melvin D. Townsend, B.S.E.E.
Captain USAF
Graduate Electrical Engineering

December 1980

Approved of public release; distribution unlimited.

Acknowledgements

This thesis is the result of a great deal of effort and could not have been accomplished without the assistance and understanding of many people. I first want to thank my fellow students, especially Capt. ''Chip'' Lutz and Capt. Lee Baker, for their assistance in microcomputer inferface problems. Next I would like to thank my thesis committee, and in particular Dr. Rustan my thesis advisor, for their assistance and counsel during this thesis. Finally, I wish to extend my sincerest thanks to my wife, Gail, without whose love and support this thesis would never have come to completion.

Contents

List of Figures

Abstract

EKG data compression techniques were investigated for potential real time implementation on an 8 bit Motorola 6800 microprocessor. Research indicated entropy reduction transform techniques such as the Fast Fourier Transform and the discrete Karhunen-Loeve Transform were not feasible for implementation on the 6800. Two redundancy reduction (RR) techniques (TOLAN and DOWER) utilizing 2nd order difference operations in conjunction with variable length encoding were studied in detail. One such RR technique (TOLAN) was fully implemented and tested with ''in vivo'' EKG data. Analysis revealed compression ratios ranging from 1.25:1 to 2.26:1. Investigation of the poor performance of the compression algorithm showed significant degradation of the 2nd order difference ''decorrelator'' due to a noisy collection environment. It was concluded that real time EKG data compression is feasible on the 6800 but that time compression techniques which store a zero value sequence counter versus the value of zero are not efficient in a high noise environment.

ANALYSIS AND PERFORMANCE EVALUATION

OF ELECTROCARDIOGRAM DATA COMPRESSION TECHNIQUES

I. Introduction

Background

Currently the USAF School of Aerospace Medicine (USAFSAM) is receiving more than thirty thousand electrocardiogram (EKG) data records a year from Air Force flight personnel worldwide. This data is presently recorded on paper strip charts for immediate medical analysis and long term storage (via microfilm).

Advances in computerized biomedical analysis has generated a need for digitized storage of the EKG waveform to allow computerized interpretation and comparison of present and past cardiographic data. As computerized diagnosis becomes more accurate, a long term digital record of historical EKG data will enable the cardiology staff at USAFSAM to identify developing heart disease before it becomes a danger to a flight crew member or his fellow crewman.

Data compression of the sampled EKG has been an area of active research since the late nineteen sixties. References (1),(7),(12),(26),(28),(29),(32),(33), and (35) are representative of the research efforts performed in the last ten years. The reasons for compressing EKG data are twofold: 1) digital storage costs are rapidly approaching

analog storage costs and ; 2) increased use of computer aided diagnosis requires large digital data bases.

In the past, EKG data compression has generally been performed at a central computer facility. The data is normally collected in a physician's office or clinic and transmitted in an analog format over a standard telephone link to the central computer. The data is then digitized and input to a computer for diagnosis and storage (in compressed form).

With the current advances in microprocessor technology, _sampling_ and _compression_ at the collection site is now a viable alternative. State of the art digital communication systems can operate at 9600 bits per second over the standard 3 kilohertz (KHz) bandwidth telephone channel. With error detection and correction protocols (Ref 27), high fidelity digital transmission of the _compressed_ EKG to the central processing center appears to be the wave of the future (Ref 21:253-254).

The Department of Defense (DOD) is currently installing a computerized EKG interpretation system at centralized US military medical centers worldwide. To collect and transmit this EKG data, remote medical clinics will use a commercial EKG ''cart'' containing a Motorola 6800 microprocessor. This EKG cart performs internal data compression, record formatting, and error protective ''channel'' encoding for _digital_ transmission to the central medical center.

As an independent study, USAFSAM has sponsored this

thesis to investigate the general field of microprocessor based EKG data compression. The results of this study are to be used for comparison against the data compression achieved in the DOD system and to create a measurement baseline to evaluate future microcomputer based EKG data compression systems.

Problem

The problem addressed by this thesis was the analysis of currently available EKG data compression algorithms for implementation in a microprocessor based computational enviroment. Additionally, a performance measure was to be developed by which differing EKG data compression techniques could be compared.

Scope

In this thesis, an EKG Data Acquistion and Analysis System (EKG-DAAS) was constructed. The EKG-DAAS collects 3 leads of an amplified EKG (i.e., Analog/Digital (A/D) dynamic range =$\pm$ 5 volts), and samples the data at a operator controlled rate between 300-700 hertz. Data is digitized at 12 bit precision and subsequently rounded to 8 bits for uniform truncation error performance (Ref 24:413-418).

The EKG-DAAS hardware was developed around a Motorola Exorciser microcomputer (6800 microprocessor) with associated A/D converter, disk memory,computer terminal, and

hard copy printer. The EKG-DAAS hardware is controlled via a software program called EKG-EXEC. EKG-EXEC is an assembly language program which performs terminal, printer, and disk input/output (I/O) operations as well as providing a supporting structure for the data compression and analysis software. Because of the lack of a high order language (e.g. FORTRAN, PASCAL, etc.) and the desire for maximum program execution speed, all programming done in this thesis is in assembly language. This has proven to be a major limitation.

Only one EKG data compression algorithm was completed and implemented on the EKG-DAAS. ''In vivo'' EKG data was taken, however, and EKG data compression performance analyzed.

General Approach

To accomplish the objectives in the problem statement, the literature was first searched for EKG compression algorithms whose implementation and execution on a 6800 microprocessor was considered feasible. As a result of this literature search and private correspondence (Ref 11 and 31), two compression routines were found. These two EKG data compression algorithms will hereafter be referred to as the Tolan and Dower methods (Ref 31,12).

With the Tolan and Dower algorithms identified, construction of the EKG-DAAS was begun by assembling the data acquistion subsystem. This data acquistion system

consists of an Analog/Digital converter and a interrupt sampling clock. The A/D was calibrated and coding of the EKG-EXEC software started. Several months of effort resulted in the EKG-EXEC software and programming of the Tolan compression algorithm initiated. Upon completion of the Tolan compression routines, the Dower algorithm was analyzed but not implemented.

Based on the results of the Tolan compression algorithm, a performance measure was formulated which compares achieved compression against an approximate maximum compression computed from the data statistics. The Dower compression algorithm was analyzed for similarity with the Tolan technique and an estimated performance figure calculated with respect to the compression measure or ''metric''.

Sequence of Presentation

Chapter 2 begins the thesis development with a general survey of the field of data compression. Data compression is shown to be divided into two subclasses (Entropy Reduction and Redundancy Reduction) and each subclass is defined. Several EKG compression techniques found in the literature were included in the ER category. These ER techniques were described and their performance advantages and limitations analyzed. Chapter 2 proceeds by describing redundancy reduction and several RR electrocardiogram compression algorithms are also analyzed. Chapter 2

concludes with the determination that implementation of the identified ER compression routines would require programming efforts beyond the scope of this thesis. Hence this thesis is limited to redundancy reduction compression.

Chapter 3 describes, in detail, the Tolan and Dower redundancy reduction algorithms whose implementation on the Exorciser was considered feasible. The Tolan algorithm is described first followed by a discussion of the Dower method. Two other EKG compression techniques, discovered during the research of this thesis, are also summarized.

Chapter 4 reviews the hardware and software configuration of the EKG-DAAS. The hardware system is described first and the specifications of the Exorciser microcomputer, A/D converter, and I/O peripherals are presented. Following the hardware description, the EKG-EXEC program is documented and the software design philosphy examined. Finally the Tolan compression module is discussed.

Chapter 5 outlines the <u>results</u> obtained using the EKG-DAAS with the Tolan algorithm. A detailed description of the experimental setup is presented along with a discussion of the <u>performance parameters</u> measured by EKG-EXEC. Finally a performance <u>metric</u> is described and the Tolan compression results compared against this metric. Chapter five ends with an analysis of how well the Dower compression technique would have performed against the performance metric. Chapter 6 concludes the thesis and

presents recommendiations for future study.

This thesis contains five appendices. Appendix A surveys basic electrocardiology and is recommended to readers unfamiliar with this subject. Appendix B is a tutorial on Information and Coding Theory and is likewise recommended to those readers unfamiliar with this field. Appendix C contains a listing of the EKG-EXEC assembly language software as well as one BASIC prcgram used on the Exorciser for data analysis. Appendix D contains data printouts of the data taken in the data collection experiment. Appendix E presents photocopies of the specification sheets for the equipment used in this thesis.

With the sequence of presention outlined, attention now turns to the theoretical section of this thesis. The first subject is data compression theory.

II. Data Compression and the EKG

Data compression is an operation in which data from an information source is ''simplified'' or ''filtered'' in a manner that produces an approximation of the original with at most some predefined amount of distortion. Some form of data compression is usually necessary when storage limitations, bandwidth requirements, or transmission channel capacity prohibit operation on the original data.

In general, data compression can be divided into two types of operations (Ref 10:4). The first operation, called entropy reduction (ER), is an irreversible transformation which reduces or compresses the data by mapping a source into an approximation of itself with a lower entropy rate. Sampling is an example of such a transformation. The second operation, known as redundancy reduction (RR) compresses the sampled data train by reducing, or eliminating, the redundancy existing in digital sequence. Since the redundant components of the data train contain no information about the source, RR is an ''exact'' data compression operation.

This chapter discusses data compression as it has been applied to the electrocardiogram (EKG). Two types of data compression strategies prevail. The first strategy involves two (or more) ER operations on the EKG data and attempts to compress the data by filtering selected components of discrete transforms. Because ER operations are irreversible, this type of data compression is sometimes

referred to as ''inexact'' compression. The second stategy performs data compression by reducing the redundancy present in the sampled EKG data train. This redundancy arises from two causes : 1) neighboring signal samples are not statistically independent and ; 2) the quantized signals amplitudes do not occur with equal probability.

Because of the speed limitations of the Motorola Exorciser microcomputer used in this thesis, only redundancy reduction algorithms were tested. This chapter, therefore, is intended as a review of the work that has been done in EKG data compression to allow comparison with the results obtained by this author.

Entropy Reduction

As is known from information theory (Ref 19), entropy {H(X)} is defined as a measure of the ''randomness'' or ''uncertainty'' of an information source. If the symbols emitted by the source are statistically independent, then the source is said to be ''memoryless'' and entropy is given by the equation:

$$H(X) = -\sum_{i=1}^{N} p_i \log_2 p_i \qquad \text{(bits)} \qquad (1)$$

where p_i represents the probability of occurrence of the ith symbol and N is the number of distinct symbols output by the source. For those ''symbols'' in a data set which occur

with zero probability, the term $p_i \log_2 p_i$ is defined equal to zer .

Immediately obvious from Eq.(1) is the observation that H(X) is defined on a discrete probability distribution. In fact, Shannon (Ref 30) defined the entropy of a continuous source as equal to positive infinity. Thus an ER transformation must, in some manner, '' discretize'' a continuous waveform into a countable set of components and attach some probability to the elements of the set.

Sampling. The clearest example of this ''discretization'' is sampling, obviously the most important operation necessary for digital signal processing. In sampling, an electrical circuit (such as an Analog-to-Digital (A/D) converter) periodically measures the value of a signal x(t) and records the data as a numeric (usually binary) number. If the signal is sampled at least twice the highest frequency component of x(t), and the duration of the sampling operation is long enough that ''aliasing'' effects (Ref 36:68-72) of ''windowing'' are negligible, then all of the ''frequency domain'' components of interest in x(t) will be preserved. Physical constraints, however, limit the accuracy of the amplitude measurement to some finite precision. Amplitude information residing below the sensitivity of the A/D converter is irretrievably lost.

As an example, let the A/D converter digitize to 8 bits. With 8 levels there are 256 possible outputs, each with a certain probability of occurrence. If the continuous

waveform fed to the A/D is the output of a stationary stochastic process with uniform statistics, then each numeric value will occur with probability of 1/256 (assuming the source max/min deviation $\geq$ the A/D dynamic range). Assuming sample-to-sample independence, the source entropy is then calculated as

$$H(X) = \sum_{256} (1/256)\log_2 256 = 8. \qquad (2)$$

Entropy has been ''reduced'' from + ∞ to 8. If the stochastic source has ''less random'' statistics (like gaussian), entropy would be reduced even futher.

All of the EKG compression routines discussed in this thesis are implemented on digital computers, hence the ER operation of sampling is always performed. In ''inexact'' EKG data compression, the next operation is another ER mapping in which the sample sequence is transformed into an alternate domain and filtered.

<u>Orthogonal Expansions and Filtering</u>. A bandlimited waveform x(t) can be expanded (over a given interval T) as a linear combination of orthonormal basis functions $\emptyset_n(t)$ (Ref 36:20-21). That is

$$x(t) = \sum_{n=0}^{\infty} a_n \phi_n \tag{3}$$

The functions $\phi_n(t)$, as $n=0,1,\ldots,\infty$, are defined as orthonormal if

$$\int_T \phi_i(t)\phi_k(t)\,dt = 1 \ , \ i=k \qquad =0 \quad \text{otherwise} \tag{4}$$

The orthonormal set $\{\phi_n(t)\}$ is called <u>complete</u> (C.O.N.) if, for any given $\varepsilon > 0$, there exists an N and a <u>finite</u> expansion

$$\tilde{x}(t) = \sum_{n=0}^{N-1} a_n \phi_n(t) \tag{5}$$

such that

$$\int_T |x(t)-\tilde{x}(t)|^2\,dt < \varepsilon \tag{6}$$

Given a complete, orthonormal set $\{\phi_n(t)\}$, then representation of any physical, noise-limited signal (over a prescribed interval T) is possible with a <u>finite</u> set of weighting coefficients $\{a_0, a_1, \ldots a_{N-1}\}$.

If k ($k<N$) terms in Eq.(5) are selectively suppressed (i.e., $a_0, a_1, \ldots a_k = 0$), *then the signal* $x(t)$ is said to be filtered of those components and a compressed representation of x(t) is produced.

EKG Filtering Compression. A great deal of work (1),(2),(26),(33),(35),(21) has been done in attempting to ''compress'' EKG data by storing, or transmitting, selective components of an orthonormal expansion. This type of compression strategy is complicated by the complex characteristics of the EKG waveform. Foremost are the limitations imposed due to the ''nonstationarity'' or variability of the EKG source, both within any given waveform and the general population as a whole.

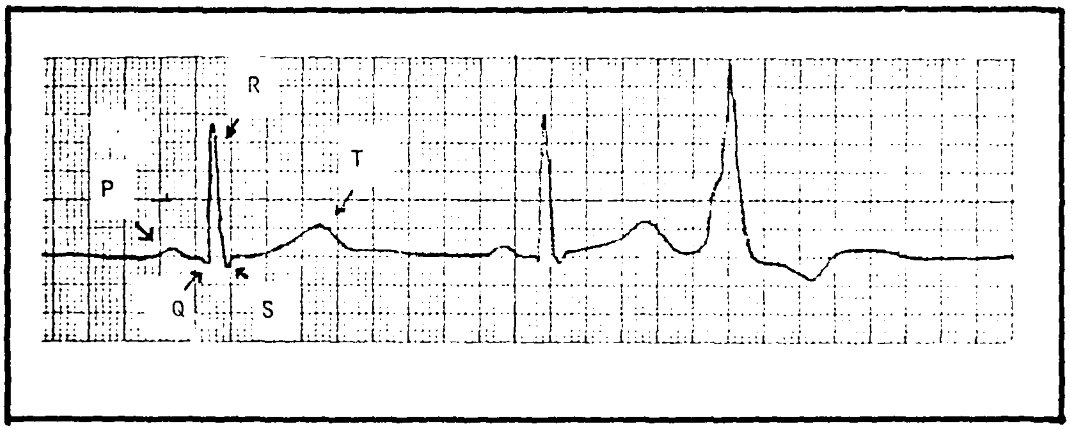


Fig 1. Typical EKG waveform with arrythmia.

During an EKG collection, the heart rate of the person undergoing testing can vary. This variation, or arrythmia, requires that a suitable reference point be established on

the P-QRS-T complex (Fig 1) around which the orthogonal expansion can be performed. Without this reference, the data falling within an expansion interval T (based purely on a constant Δt) will appear to be ''almost'' random. If the ''filtering'' used to effect compression is based on an expected waveform within the transform window, improper registration will cause severe degradation in compression efficiency. The above situation is known as the ''epoch'' problem and is discussed by Womble (Ref 35:703).

Assuming that the expansion interval is aligned properly on some feature of the EKG (e.g. R wave), then the problem of variable P,QRS, and T complexes must be considered.

If the individual under test has a heart disease, then in many cases the P,QRS,and T segments of that person's EKG may vary significantly from the population ''norm''. Again, if the compression filtering is based on an assumed, or ''normal'' waveform, then significant reduction in compression efficiency can be expected.

These problems, among others, require that the EKG ''filtering'' compressor be robust enough to handle the variations possible within the waveform. With these limitations in mind, discussion will now proceed to compression transforms.

Transforms. Two major types of discrete filtering compression strategies, as applied to EKGs, will now be discussed. These are the Fast Fourier Transform (FFT) and

the discrete Karhunen-Loeve Transform (DKLT). These two techniques are representative of current research into EKG filtering compression and will serve as good basepoints against which to compare the redundancy reduction compression techniques outlined in chapter 3.

Fast Fourier Transform. The fast Fourier transform is a computationally efficient algorithm for calculating the discrete Fourier transform (DFT). The following discussion involves the DFT, but calculation by the FFT is implied.

Data compression using the DFT is achieved by zonal filtering in which ''zones'' of the transform sequence are selectively discarded. Usually these zones are defined by a ''cutoff frequency'' f_c and only those components less than or equal to f_c are saved. Data reconstruction is performed by computing the inverse DFT (FFT) on the filtered sequence and replacing the filtered components with zeroes. Using the symmetry relationships described in Oppenheim and Shaffer (Ref 24:103-105), then the transform sequence for N=8 is:

$$X(k) = \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} = \begin{bmatrix} R(0) + jIm(0) \\ R(1) + jIm(1) \\ R(2) + jIm(2) \\ R(3) + jIm(3) \\ R(4) + jIm(4) \\ R(3) - jIm(3) \\ R(2) - jIm(2) \\ R(1) - jIm(1) \end{bmatrix} \tag{7}$$

where R is the real part, Im the imaginary part of the complex number X, and $j=\sqrt{-1}$.

Zonal filtering takes advantage of the symmetry in Eq.

(7) by saving those positive frequency components less than or equal to a certain cutoff frequency f_c. IF f_c were chosen as equal to X(2), then only X(0),X(1), and X(2) would be saved. On reconstruction, X(3),X(4),and X(5) would be set equal to zero (i.e. they have been filtered out) while X(6) and X(7) would be recreated from X(1) and X(2) using symmetry. The inverse DFT (FFT) now produces a ''filtered'' approximation of x(n).

EKG data compression via filtered FFT spectra has been studied by Womble (Ref 35) and the TRW corporation (Ref 33). In both studies the distortion criterion used in establishing f_c was visible reproducibility with no detectable distortion.

In the TRW study, compression ratios (defined as bits in : bits out) from 2:1 to as high as 17:1 were obtained using zonal filtering. The high compression ratios were measured using highly rhythmic EKGs taken from an individual (an astronaut) in a low noise environment. This extraordinary compression ratio is far from ''normal'', however. Womble (Ref 35) has shown that, on the average, 40-80 terms of a 512 sample FFT (sampled at 500 Hz) are necessary to reproduce the EKG with acceptable visual distortion. Since the FFT requires storage of 2 numbers per term (real and imaginary components) then, in general, FFT zonal filtering compression ratios of 5:1 to 3:1 are more common.

The FFT is often used in digital signal processing

because of its speed and representation in the frequency domain. As will be seen from the data in chapter 5, compression ratios of 5 : 1 are usually higher than those obtainable by redundancy reduction techniques. Even so, the FFT is not the ''optimal'' transform for representing a sample sequence.

<u>Discrete Karhunen-Loeve Transform</u>. Because all of the filtering compression techniques are ER transformations, distortion upon reconstruction is inevitable. One would like a performance measure against which this distortion could be evaluated.

One common measure of performance is the <u>mean square error</u> defined by the equation:

$$e(M)=E\{(\underline{X} - \underline{\tilde{X}})^2\} \tag{8}$$

where E is the statistical expectation operator, $\underline{X}$ is a discrete signal composed of N sample values, and $\underline{\tilde{X}}$ is the estimate of $\underline{X}$ via some orthogonal coordinate system. Given that a representation of $\underline{X}$ is desired with less than N components, Ahmed and Rao (Ref 2:200-203) have shown that the DKLT is the <u>optimum</u> transform to minimize mean square error.

The DKLT can be described as follows: Let the orthonormal transform matrix [T] be defined as

$$[T]' = [\,\emptyset_1, \emptyset_2, \ldots \emptyset_N\,] \qquad \{[T]'=[T] \text{ transpose}\} \tag{9}$$

and $\emptyset_i$ are N dimensional, real valued basis vectors. Hence a transform vector $\underline{Y}$ can be created by

$$\underline{Y} = [T]\,\underline{X} \tag{10}$$

where $\underline{X} = [x_1, x_2, \ldots x_N]$; $\underline{Y} = [y_1, y_2, \ldots y_N]$. Since the $\emptyset_i$ are orthonormal, then

$$\underline{X} = y_1\emptyset_1 + y_2\emptyset_2 + \ldots + y_N\emptyset_N = \sum_{i=1}^{N} y_n\emptyset_n \tag{11}$$

The goal is to optimally represent $\underline{X}$ by a subset $\{y_1, y_2, \ldots, y_M\}$ where $M < N$. If the remaining N−M terms are represented by the constants b_i, *then an estimate* $\tilde{\underline{X}}$ is defined such that

$$\tilde{\underline{X}} = \sum_{i=1}^{M} y_i\emptyset_i + \sum_{i=M+1}^{N} b_i\emptyset_i \tag{12}$$

An error vector, $\Delta\underline{X}$ is now created where

$$\Delta\underline{X} = (\underline{X} - \tilde{\underline{X}}) = \sum_{i=M+1}^{N} (y_i - b_i)\emptyset_i \tag{13}$$

Now the mean square error defined in Eq.(8) can be redefined

as

$$e(M)=E\{(\Delta\underline{X})'(\Delta\underline{X})\}$$

$$=E\{\sum_{i=M+1}^{N}\sum_{j=M+1}^{N}(y_i-b_i)(y_j-b_j)(\emptyset'_i\emptyset_j)\}$$

$$=\sum_{i=M+1}^{N}E\{(y_i-b_i)^2\} \tag{14}$$

Ahmed and Rao (Ref 2:202) show that by minimizing Eq.(14)

$$e(M) = \sum_{i=M+1}^{N}[\ \emptyset'_i E\{(\ \underline{X} - \bar{\underline{X}}\ )(\ \underline{X} - \bar{\underline{X}}\ )\}'\emptyset_i] \tag{15}$$

where $\bar{\underline{X}}$ is the mean of $\underline{X}$.

The expectation in Eq. (15) is recognized as the covariance of $\underline{X}$, hence Eq. (15) becomes

$$e(M) = \sum_{i=M+1}^{N}(\ \emptyset'_i[K_x]\emptyset_i) \tag{16}$$

where $[K_x]$ is the covariance matrix of $\underline{X}$.

By minimizing Eq.(16), Ahmed and Rao (Ref 2:200-205)

show that the othornormal functions $\emptyset_i$ become the eigenfunctions of the covariance matrix $[K_x]$ and the mean square error becomes

$$e(M) = \sum_{i=M+1}^{N} \lambda_i \tag{17}$$

where the λ_i are the eigenvalues of $[K_x]$. By expanding $\underline{X}$ with the eigenfunctions corresponding to the M largest eigenvalues, then $\underline{X}$ is filtered of N-M components with minimum mean square error. This is the discrete Karhunen-Loeve transform.

The DKLT has been studied by Ahmed (Ref 1) and Womble (Ref 35). Both studies compared the performance of EKG data compression using the DKLT against other orthogonal transforms.

Ahmed's tests utilized canine EKGs and clearly showed the optimality (in the ''mean square sense'') of the DKLT. Ahmed, however, deemed the DKLT to be too complex for practical implementation, and proceeded with the development of EKG compression strategies using suboptimal transforms. The details of these suboptimal expansions will not be given here, but the reader is referrenced to Ahmed's paper (Ref 1).

Womble, on the otherhand, demonstrated that the DKLT is not to difficult to implement. In his experiments, Womble

took 3 lead vectorcardiogram (VCG) data (see appendix A), hereafter referred to as X,Y,Z, and first transformed the data into a different, orthogonal coordinate system (U,V,W). The transformation was determined by solving for the eigenvalues of the 3 X 3 matrix:

$$S = 1/N \sum_{i=1}^{N} \begin{bmatrix} X(i) \\ Y(i) \\ Z(i) \end{bmatrix} \begin{bmatrix} X(i), Y(i), Z(i) \end{bmatrix} \tag{18}$$

where N was chosen equal to 200 and X(i),Y(i),Z(i) are the X,Y,Z components of the VCG in the Frank (Ref 14) coordinate system.

Next an ''average'', or mean, heartbeat was calculated using 900 patients and the data (in the eigenvector coordinate system) was subtracted from the mean forming a ''patient'' vector ρ. The vector ρ is defined as

$$\rho = \begin{bmatrix} u(1)-u_m(1) \\ u(2)-u_m(2) \\ \cdot \\ \cdot \\ u(N)-u_m(N) \\ v(1)-v_m(1) \\ v(2)-v_m(2) \\ \cdot \\ \cdot \\ v(N)-v_m(N) \\ z(1)-z_m(1) \\ z(2)-z_m(2) \\ \cdot \\ \cdot \\ z(N)-z_m(N) \end{bmatrix} \tag{19}$$

This patient vector ρ is expanded using the eigenfunctions

of the matrix

$$L = 1/N \sum_{i=1}^{N} (\rho_i)(\rho_i)' \tag{20}$$

where N is now a large number of patients being ''averaged'' (up to 300 reported in Ref 33). The eigenvectors of this tremendous matrix are calculated (once for all time), and then M (M << 200) are used to expand the data vector ρ.

The above approach has allowed accurate representation of EKG sample sequences, properly aligned on the heartbeat, with an M=20 DKLT coefficients per lead per second (Ref 35). At Womble's sample rate of 250 Hz, this represents a compression ratio of over 12 : 1 . With the most significant eigenfunctions stored from the solution of Eq. (20), then the compression of digitized EKG data by filtering the DKLT expansion on ρ is approaching feasibility on a microcomputer.

The preceeding discussion was intended as a basic tutorial on some of the entropy reducing (ER) transformations currently under test for EKG data compression. As stated earlier, ER operations are ''inexact'' because some ''information'' is always discarded. By discarding data efficiently, zonal filtering can achieve compression ratios larger than those obtained using the redundancy reduction techniques implemented in this thesis.

Redundancy Reduction

Redundancy reduction is an operation which performs data compression by indentifying and removing the redundant components of a digital sequence. This redundancy exists for two reasons : 1) the sample points are not statistically independent and ; 2) the quantized amplitude values of the sample train do not occur with equal probability. The redundant components of the data sequence carry no ''information'' (see appendix B), hence their removal does not affect the ''message'' content of the data.

This section of chapter two discusses redundancy reduction (RR) operations which have been applied to the EKG. This is done because the compression algorithms examined in this thesis are of the RR type, and the background given here will aid in the descriptions given in chapter three.

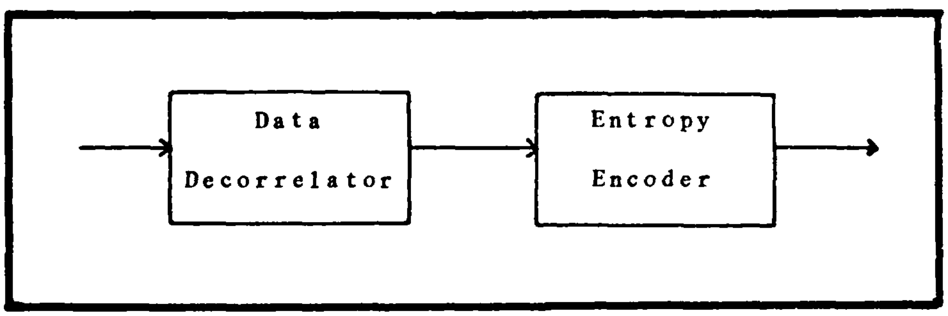


Fig 2. Typical EKG Redundancy Reduction Data Compressor.

The sampled EKG contains redundancy resulting from both

sample-to-sample correlation and unequal amplitude probability. As is illustrated by Figure 2, each of these redundancy components can be reduced by a different operator.

The decorrelator, as inferred by its title, transforms the data in a manner which decorrelates the data stream. The decorrelated residual sequence is then efficiently compressed by means of the entropy encoder.

To reduce the rendundancy arising from intersample correlation, two basic approaches are used. The first approach utilizes linear predictor/interpolators (Ref 28,33) to produce an information bearing ''residual '' sequence. The second method generates the residual sequence by taking successive differences on the data stream. As will be shown, these residual sequences carry <u>all</u> of the <u>source information</u> with minimum intersample redundancy. The predictor/interpolator method is discussed first.

<u>Predictors and Interpolators</u>. A predictor estimates the next sample value of a sequence (i.e. x_n) based on a linear combination of k past samples. That is

$$\tilde{x}_n = \sum_{i=1}^{k} a_i x_{n-i} \tag{21}$$

where a_i are coefficients chosen to minimize the <u>mean square error</u> $\sigma_e^2 = E\{(X-\tilde{X})^2\}$ between the sample sequence X_n and the

predicted sequence $\tilde{X}_n$. An error , or residual, sequence e_n can then be formed where

$$e_n = X_n - \tilde{X}_n \tag{22}$$

thus

$$X_n = \tilde{X}_n + e_n \tag{23}$$

Conceptually, this technique decomposes a sample value into a part which is correlated with k past sample values (i.e. the redundant component) and a part which is uncorrelated with them. Linear mean square estimation theory (Ref 24:385-430) shows that the uncorrelated part (e_n) may be retained alone with <u>no</u> loss of information.

In their research in EKG data compression, Ruttimann and Pipberger (Ref 28:616) prove that since $E\{e_n\}=0$, then the mean square error σ_e^2 is equal to the variance of e_n . With a second order predictor (k=2), Ruttimann and Pipberger have demonstrated a variance reduction (σ_x^2/σ_e^2) of over 25 : 1. This reduction in variance implies that the EKG residual sequence e_n is much more tightly clustered around a given mean than is the original sequence X_n . This clustering, as will be discussed later, enhances data compression by means of <u>entropy</u> ,or <u>source encoding</u>.

In an analogous way, interpolators estimate a value of x_n . In the case of the interpolator, however, the estimate

of x_n consists of a linear combination of past <u>and</u> future samples. That is

$$\tilde{x}_n = \sum_{i=1}^{k} a_i x_{n-i} + \sum_{i=1}^{m} b_i x_{n+i} \tag{24}$$

where k past and m future values are used. As with the predictor, the coefficients a_i and b_i are again chosen to minimize the mean square error. As was the case with the second order predictor, Ruttiman and Pipberger (Ref 28:617) have found that a <u>second order</u> interpolator (k=1,m=1) yields EKG residual sequences e_n with the most significant variance reductions (greater than 31 : 1).

EKG data compression utilizing predictors/interpolators have been studied by the TRW Corporation (Ref 33), as well as Ruttimann and Pipberger (Ref 28). In both cases, predictors/interpolators of order 2 seem to prevail. As is shown by the two groups above, second order systems have the smallest residual sequence variance hence are most ameanable to entropy encoding. An alternate approach used to reduce intersample redundancy is by means of difference operations.

<u>Difference Reduction</u>. In difference reduction, the residual sequence is formed by taking successive differences of the sample data train. Because there is no multiplication (as in Eq.(24)), the difference operation is inherently a

simpler procedure than is prediction or interpolation. Difference sequences, however, are not optimized with respect to mean square error and less ''efficient'' redundancy reduction (decorrelation) occurs.

An example of an EKG difference reduction compressor is illustrated in Figure 3. This technique was implemented by Cox and Ripley (Ref 7) and tested against a data base of 45 patients. Figure 4 shows that Cox and Ripley's second difference decorrelator produced a sharply peaked relative frequency distribution. This ''peakedness'' or clustering of the residual sequence permits efficeicnt entropy encoding. A variant of the difference reduction above is time compression.

Time Compression. EKG time compression, as implemented by Dower,Berghofer, and Stewart (Ref 12,29), uses a second order difference reduction but instead of keeping the value of zero (the most common), a run length counter (Δt) is kept. This run length counter measures the number of repetitive $\Delta^2 X$ sequence terms equal to zero. By saving (then encoding) only those $\Delta^2 X \neq 0$,and the Δt between the nonzero second differences, data compression can be realized.

Time compression is the method that has been implemented and studied extensively by this author. Chapter 3 discusses two different RR compression algorithms which both use second order time compression for the decorrelator. For now, dicussion is focused on the second

operation identified in Figure 2, the entropy encoder.

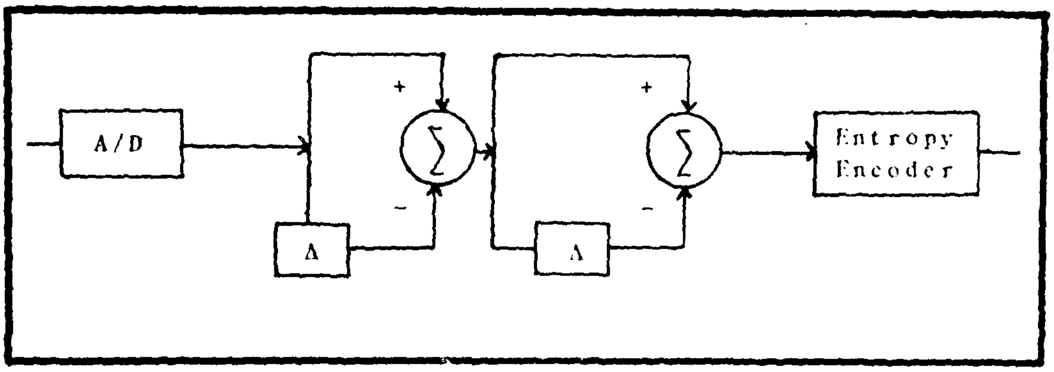


Fig 3. Second Order Difference Operator (from Ref 7:336).

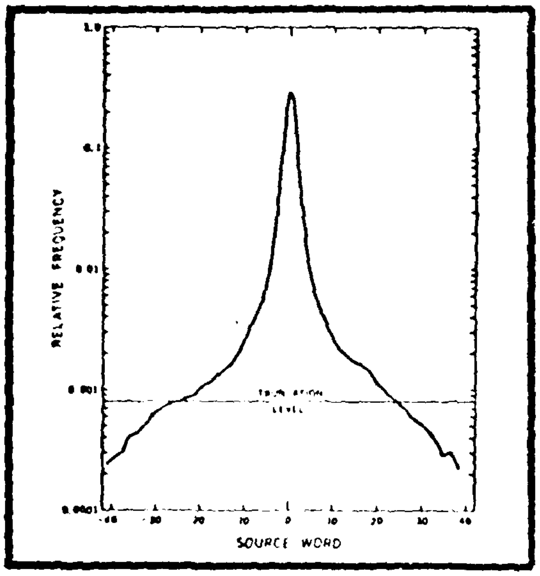


Fig 4. Relative Frequency of 2nd Order Difference Operator (Ref 7:336)

Entropy Encoding. Entropy, or source, encoding operates on the remaining redundancy in the residual sequence e_n resulting from unequal source symbol probability. How well e_n can be compressed is given by Shannon's noisless source coding theorem (Ref 30) which states that the average number of binary symbols per source output can be made to approach the entropy of the source and no less. Unfortunately, the EKG ''source'' has memory, hence the entropy of this source will be less than that which would be calculated from Eq.(1). In general, the true value of a nonstationary, memory source is difficult (or impossible) to calculate.

To circumvent this ''memory'' problem, the assumption is made in the literature (Ref 29,12, 7,35) that the ''uncorrelated'' nature of the residual sequence e_n approaches independence, hence the entropy of e_n as calculated by Eq.(1) is a good bound on the possible compression. Since the variance of e_n is sharply peaked around zero, entropy encoding via variable length coding (VLC) appears attractive.

In variable length coding, those values of e_n which occur most often are assigned the shortest code words (i.e. fewest code symbols per source symbol). As example, assume that both the source symbol alphabet and the code alphabet are binary. The values in the sequence e_n are the result of algebraic operations on fixed length numeric sample values and are hence fixed length binary numbers (e.g. L=8). The variable length coder maps these fixed length numeric values

(binary) into *uniquely decodable* (UD) binary code words whose bit length is a function of the frequency of occurrence of the values in e_n .

Coding theory (Ref 19:237-248) shows that, in principle, it is possible to have variable length codes with an *average bit length* ($\underline{l}$) equal to the entropy of source. Hence with the right VLC, a compression ratio of L/ $\underline{l}$ is achievable.

A common VLC used in EKG data compression research (Ref 7,28) is the Huffman code. Huffman, in 1952, developed an algorithm (Ref 16) for generating the optimal UD code (assuming stationary, memoryless source). This code is in a class of UD codes called *prefix codes* in which no code word is the prefix of any other. Unfortunately, a codeword must be assigned to every possible symbol which occurs, regardless of how infrequently. This means that although the average code word bit length approaches the entropy of the source, the longest code word can be substantially larger. In a straight Huffman code, these ''long'' code words can induce severe problems due to ''buffer overflow'' (Ref 17).

For practical implementation in EKG data compression, Ruttimann and Pipberger (Ref 28) and Cox and Ripley (Ref 7) constructed a modified Huffman code. In this modified code, the residual sequence source words are partitioned into a frequent set and an infrequent set (known as ''else''). A Huffman code was then formed with all of the residual words

in the first set plus a special code word used as a prefix for any source word from the infrequent set. The prefix, when it occurs, is followed by a fixed length suffix which contains the value of the infrequent source word. The probability that ''else'' will occur must be kept small enough to mantain the efficiency of the truncated Huffman code.

Using the modified Huffman code in conjunction with a second order interpolator, Ruttimann and Pipberger (Ref 28) have attained compression ratios as high as 9 : 1. To get this ratio, however, significant digital signal processing was performed on the ''raw'' 8 bit data. This preprocessing involved digital filtering for noise reduction and Lagrange interpolation/decimation to produce an effective 200 Hz sample rate. Nonetheless, this represents a significant approach to the compression ratios attainable by the discrete Karhunen-Loeve transform mentioned previously.

Although optimal, Huffman codes are not the only VLC used in EKG data compression. Two different codes will be described in chapter three, one of which (Dower code) may possibly be ''more optimal'' than Huffman <u>in the compression of quantized EKGs</u>.

Summary

This chapter has reviewed the theory of data compression and how this theory has been applied to the electrocardiogram. It was shown that data compression could

be partitioned into two types of operations: 1) entropy reducing transformations which map a data source into an approximation of itself with a lower entropy rate and ; 2) redundancy reduction operations which compress by removing redundancy resulting from sample-to-sample dependence and unequal source symbol probabilities.

The first operation was applied to the EKG via ''filtering'' of orthogonal expansions of the digitized EKG. Filtering transforms, especially the discrete Karhunen-Loeve transform, were shown to be very efficient ''compressors'' if the loss of the filtered components were tolerable. If this loss was not acceptable, then redundancy reduction operations are used.

In redundancy reduction techniques, the EKG is first *processed* by passing the digitized data through a decorrelator which reduced the redundant component caused by source symbol dependence. This decorrelator could be implemented with predictor/interpolators or difference operations. Next the data is encoded via entropy encoding operations, usually utilizing variable length codes. It was finally shown that a second order interpolator, followed by an ''optimal'' Huffman encoder could achieve compression ratios which approach those obtained by the discret Karhunen-Loeve transform. Both of these last two techniques, however, require substantial computational overhead.

III. Tolan and Dower EKG Compression Techniques

Two EKG data compression algorithms are studied in detail in this chapter. The first compression technique, hereafter referred to as the Tolan method, was conceived by Dr. Gil Tolan,MC from the USAF School of Aerospace Medicine (USAFSAM), Brooks AFB, Texas. The second compression approach, referred to as the Dower method, was developed by Mr. Roger Dower and Mr. Dave Berghofer from Shaughnessy Hospital, Vancouver, B.C., Canada. Both of the EKG compression procedures are redundancy reduction operations using *time compression* for decorrelation and variable length codes for entropy encoding.

This author had originally intended to implement and test both the Tolan algorithm and the Dower algorithm on the Motorola microcomputer (See Chapter 4). Unfortunately, difficulties with hardware failures and insufficient software tools (i.e., a high order language) prevented implementation of the Dower algorithm. Nonetheless, this chapter compares the design of both the Tolan and the Dower algorithms and illustrates their differences as well as their similarities. Based on the results of the Tolan compression approach (chapter 5), this comparison will illustrate the potential performance of the Dower compression stategy in a real time microcomputer enviroment. Before continuing with the Dower and Tolan algorithm descriptions, a short digression will be made.

In both the Tolan and The Dower compressors, a second

difference operation is used to produce an information bearing sequence with reduced component-to-component dependence. Why the second difference operation results in the lowest correlation is a question for closer scrunity.

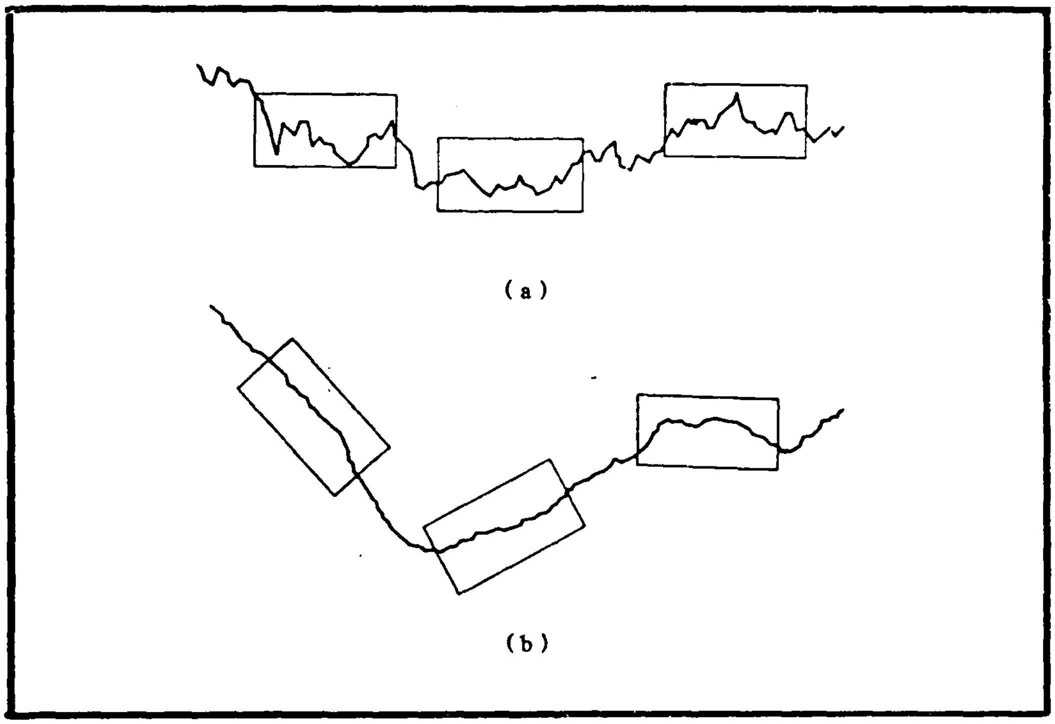


Fig 5. Nonstationary waveforms with random level and slope. (From ref 6:91)

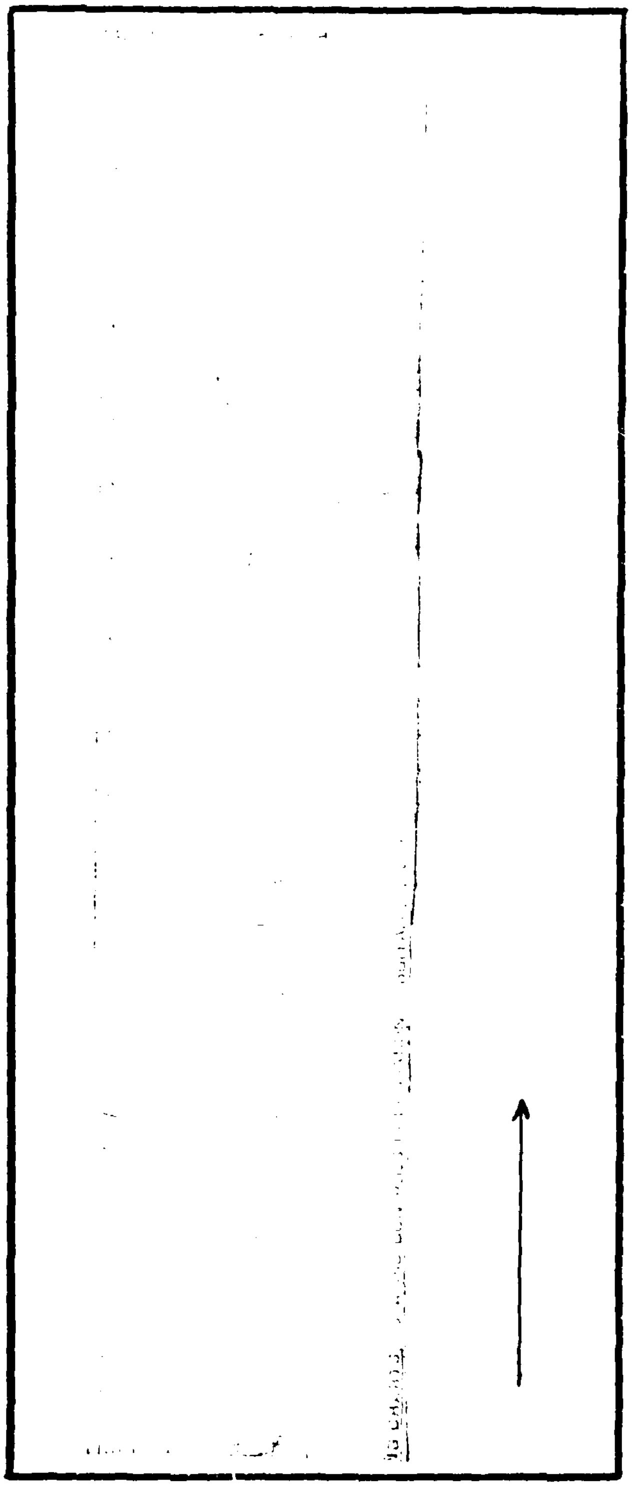
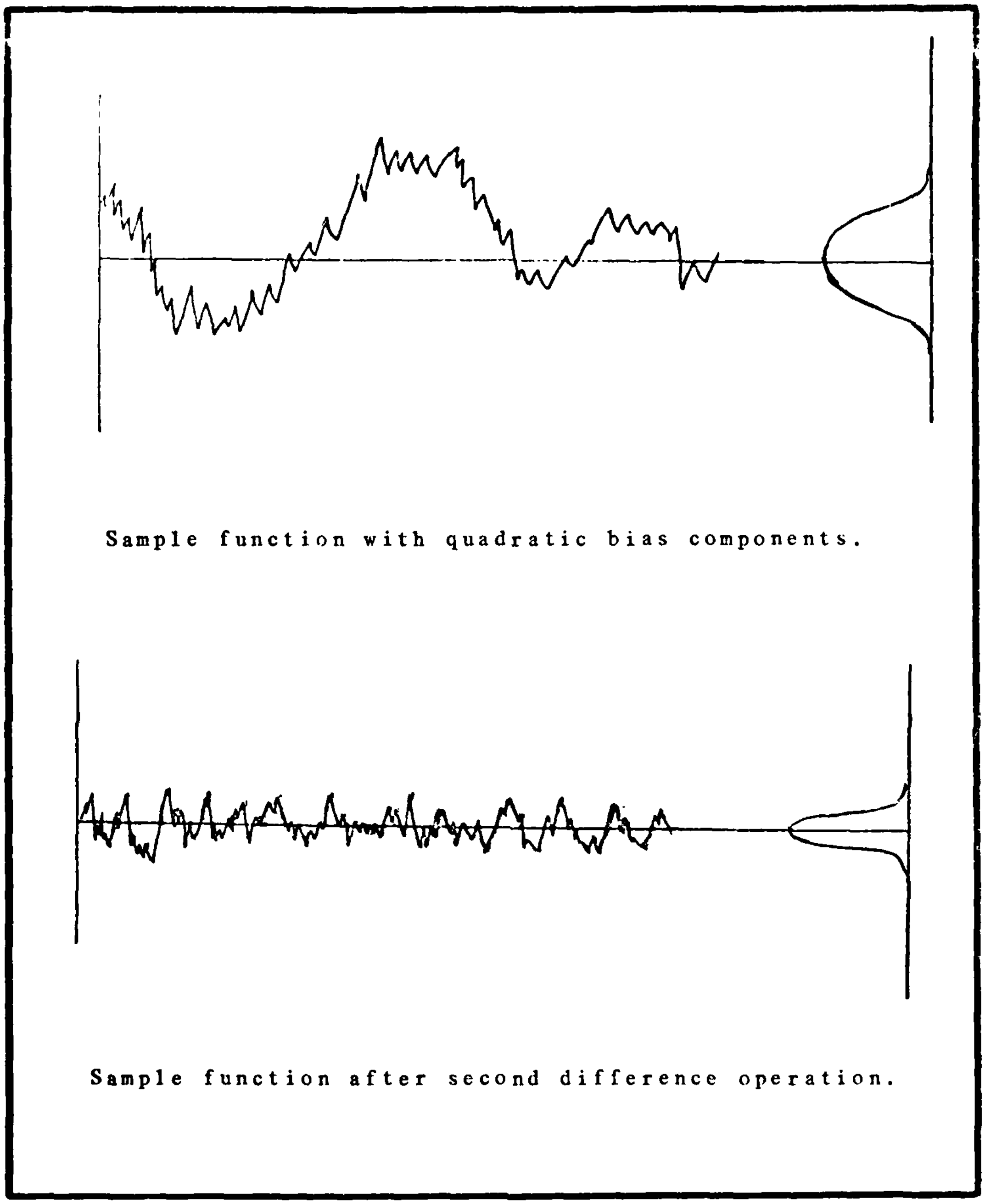


Fig 6. EKG waveform with moving average slope.

A _nonstationary_ process (like the EKG) exhibits a wide amplitude distribution due to the random ''wander'' of the waveform as shown in Figure 6. This wander, or ''moving average'', can be induced by patient variation, EKG apparatus drift, or both. Time series analysis (Ref 6) shows that a nonstationary process which has sample functions which are ''locally stationary'' or homogenous (Figure 5) can be represented by a process model which calls for the _d'th difference of the process to be stationary_. The proof of this assertion is given by Box and Jenkins (Ref 6:85-125).

If the nonstationary process sample sequence (time series) exhibits a random level as illustrated in Figure 5a, then a first difference operation will remove this ''moving average'' and force the resulting difference sequence to be centered around zero. If the waveform exhibits a random slope, as shown in Figure 5b, then a second difference operation will remove this quadratic ''bias'' with a corresponding reduction in amplitude distribution variance (Figure 7.) Comparison of Figures 5 and 6 shows that an EKG trace can ''look'' similar to the example in Figure 5b.

From the theory of stochastic processes (Ref 9:330-331), it is known that the expected value of a sample mean obtained by sampling a wide-sense stationary random process along a sample function in time is equal to the constant mean value of that random process.



Sample function with quadratic bias components.

Sample function after second difference operation.

Fig 7. Simulated variance reduction via second differencing.

In addition, the variance of the sample mean is inversely proportinal to the number of samples taken when the samples are pairwise uncorrelated. Another fact about widesense stationary processes is that their autocorrelation functions are dependent only on the time difference between observation of the sample sequence and that $R(t_1 - t_2) \leq R(0)$ for $t_1 - t_2 \neq 0$. Since the second difference is the first ''difference'' which can be modelled as coming from a ''stationary'' process, and the maximimum correlation occurs for zero time difference, then the correlation must be reduced for adjacent sequence values in the second difference operation.

This hueristic argument is far from complete and does not explain why the third difference exhibits worse behavior than the second difference. The EKG waveform is a complex function from a complex source and higher order effects could begin to dominate with the third difference operation. Further analysis of this anomaly is left for future study.

The remainder of this chapter is organized in the following manner. First the Tolan algorithm is described, followed by a description of the Dower procedure. Next a short synopsis will be made of three other EKG data compression algorithms uncovered during the research of this thesis.

Tolan EKG Data Compressor

The Tolan EKG data compression algorithm is a redundancy reduction procedure which processes a three lead EKG (VCG) and produces a compressed, digital output. This digital output sequence could subsequently be channel encoded for ''errorless'' transmisson (Ref 27 and Appendix B) or stored for later retrieval and reconstruction.

As was the case for the RR techniques described in chapter 2, the Tolan compressor is subdivided into a data decorrelator and an entropy encoder. The decorrelator is discussed first.

Tolan Decorrelator. The Tolan decorrelator is a second order difference reduction operation which utilizes a time compression approach to form a decorrelated output sequence. A second order system was chosen based on the experimental results of Dower and Berghofer (Ref 12) and Cox and Ripley (Ref 7) which showed maximum compression gain with a second order difference operation. The Tolan decorrelator works on a three lead EKG (VCG) signal set, assumed to be sampled at a constant rate. The data compression is achieved in real time between successive samples.

The Tolan decorrelator algorithm is defined in Figure 8. Close examination of the algorithm in Figure 8 reveals that the second difference data is stored only if : 1) any of the three Δ^2 values are nonzero or ; 2) if the Δt counter

records more than 127 repetitive cases where $\Delta^2x=\Delta^2y=\Delta^2z=0$. Rule 1 was chosen due to the high degree of correlation between the leads of an FKG (i.e. when one lead is changing, so are the others). The capacity of the time counter in rule 2 was arbitrarily chosen to be sufficient to record the long quiescent periods which occur in the EKG (see Figure 1) and short enough to be efficiently stored.

Step 12 of the Tolan algorithm in Figure 8 calls the variable length encoding subroutine (Figure 10) which encodes the Δ^2 terms calculated by the second difference decorrelator. This encoding procedure is the next subject to be discussed.

<u>Tolan Entropy Encoder</u>. The Tolan code is an uniquely decodable variable length code which stores the Δ^2 values as a contiguous sequence of binary 1's. The length of this ''run'' of binary 1's is equal to the magnitude of the Δ^2 term. To delineate between the ''runs'', binary 0's are used as codeword delimiters. Since the second difference has both negative and positive values, a sign bit (0 for positive, 1 for negative) immediately follows the 0 bit delimiter. The three values $\Delta^2x, \Delta^2y, \Delta^2z$ are encoded and stored sequentially followed by a delimited, 7 bit, <u>uncoded</u> Δt value. A Δ^2 value of <u>zero</u> is indicated by 3 successive 0 bits.

1: $\Delta x=0,\ \Delta y=0,\ \Delta z=0,\ \Delta t=1$

2: $x(i)=(a/d\ ch\ 0), y(i)=(a/d\ ch\ 1), z(i)=(a/d\ ch\ 2)$

3: $mem(0)=x(i), mem(1)=y(i), mem(2)=z(i)$

4: $\tilde{x}(i+1)=x(i)+\Delta x,\ \tilde{y}(i+1)=y(i)+\Delta y,\ \tilde{z}(i+1)=z(i)+\Delta z$

5: if ready for next sample then GOTO 6 else GOTO 5

6: $i=i+1, x(i)=(a/d\ ch\ 0), y(i)=(a/d\ ch\ 1), z(i)=(a/d\ ch\ 2)$

7: $\Delta^2 x=x(i)-\tilde{x}(i),\ \Delta^2 y=y(i)-\tilde{y}(i),\ \Delta^2 z=z(i)-\tilde{z}(i)$

8: if $\Delta^2 x\neq 0$ or $\Delta^2 y\neq 0$ or $\Delta^2 z\neq 0$ then GOTO 11 else GOTO 9

9: $\Delta t=\Delta t + 1$

10: if $\Delta t \leq 127$ then GOTO 4 else GOTO 12

11: $\Delta x=\Delta x + \Delta^2 x,\ \Delta y=\Delta y + \Delta^2 y,\ \Delta z=\Delta z + \Delta^2 z$

12: go subroutine coder $\{\Delta^2 x, \Delta^2 y, \Delta^2 z, \Delta t\}$

13: if memory is full then STOP else GOTO 14

14: $\Delta t=1$

15: GOTO 4

where

$x(i), y(i), z(i)$ = sampled, 8 bit precision, EKG data

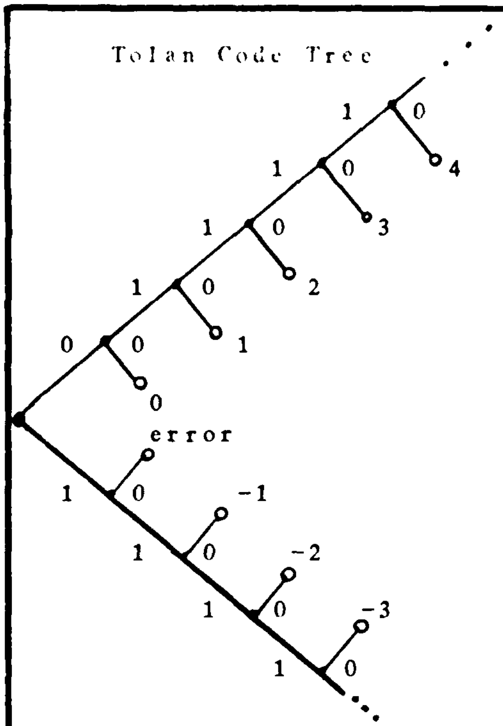
$\Delta x, \Delta y, \Delta z$ = first difference $\{\Delta x(n)=x(n)-x(n-1)\}$

$\Delta^2 x, \Delta^2 y, \Delta^2 z$ = second difference $\{\Delta^2 x(n)=\Delta x(n)-\Delta x(n-1)\}$

Δt = time difference between nonzero Δ^2 values

$\tilde{x}(i+1), \tilde{y}(i+1), \tilde{z}(i+1)$ = next predicted data points

Fig 8. Tolan Collection and Decorrelation Algorithm.



Tolan Code

0:	0 0
error:	1 0
1:	0 1 0
-1:	1 1 0
2:	0 1 1 0
-2:	1 1 1 0
3:	0 1 1 1 0
-3:	1 1 1 1 0
4:	0 1 1 1 1 0

Fig 9. Tolan Code and Code Tree (8 smallest codewords).

As an example of Tolan encoding, let $\Delta^2 x=3$, $\Delta^2 y=-1$, $\Delta^2 z=0$, $\Delta t=13$. With these values, then the code string generated is:

D S V V V D S V D S D T T T T T T T D

0 0 1 1 1 0 1 1 0 0 0 0 0 0 1 1 0 1 0

where

D = Delimiter bit

S = Sign bit

V = Value bit

T = Time bit

Examination of the above sequence shows that the Δ^2

codewords are prefix codes as shown in the tree diagram of Fig. 9. To decode the sequence correctly, however, it is imperative that codeword synchronization be maintained in order to determine when the coded Δ^2 values end and the uncoded, 7 bit Δt variable starts. This loss of synchronization is detected by the ''error'' state shown in the code tree of Figure 8. The decoder which implements the code tree in Figure 9 will not be discussed here but is listed in Appendix C.

In contrast with the Huffman code (see chapter 2 and Ref 16) and the Dower code to be discussed, the Tolan code is <u>not</u> constructed using the apriori knowledge of the source word (i.e. $\Delta^2 x, \Delta^2 y, \Delta^2 z$) relative frequency of occurrence. This means that the Tolan code will only produce bit compression (bits out/bits in/< 1) if the Δ^2 values are <u>sharply peaked</u> around a mean of zero such that few code words exceed the 8 bit, fixed length value of the Δ^2 terms. As will be shown in chapter 5, the second order difference decorrelator does produce such a sharply peaked relative distribution.

The algorithm which implements the Tolan variable length coder is shown in Figure 10. This algorithm works in conjunction with the Tolan decorrelator in Figure 8.

```
1:  entry (Δ²x,Δ²y,Δ²x,Δt)
2:  cnt <- 0
3:  reset current memory bit, increment bit pointers
4:  if end of memory,set eom flag and RETURN else GOTO 5
5:  cnt <- cnt + 1
6:  if cnt=1 then tvar <- Δ²x
7:  if cnt=2 then tvar <- Δ²y
8:  if cnt=3 then tvar <- Δ²z
9:  if cnt ≥ 4 then GOTO 18 else GOTO 10
10: if tvar ≥ 0 then GOTO 11 else GOTO 12
11: reset memory bit, increment bit counters and GOTO 13
12: set memory bit, increment bit counters
13: if end of memory,set eom flag and RETURN else GOTO 14
14: if tvar=0 then GOTO 5 else GOTO 15
15: set memory bit, increment bit counters
16: if end of memory,set eom flag and RETURN else GOTO 17
17: tvar <-  tvar - 1 and GOTO 14
17: store 7 bit Δt counter to memory, update bit pointers
18: if end of memory,set eom flag and RETURN else RETURN
```

Fig 10. Tolan Variable Length Encoder Algorithm.

```
 0: START OF DOWER DATA COLLECTION AND DECORRELATION

 1: FLAG=TDT=DT=DX1=DY1=DZ1=DX2=DY2=DZ2=DDX=DDY=DDZ=CNT=0

 2: If sample interrupt detected then GOTO 3 else GOTO 2

 3: X(n) ← A/D CH 0, Y(n) ← A/D CH 1, Y(n) ← A/D CH 2

 4: CNT=CNT + 1

 5: If CNT =1 then GOTO 6 else GOTO 7

 6: X(n-1) ← X(n), Y(n-1) ← Y(n), Z(n-1) ← Z(n) and GOTO 2

 7: If CNT =2 then GOTO 8 else GOTO 13

 8: DX1 ← X(n)-X(n-1),DY1 ← Y(n)-Y(n-1),DZ1 ← Z(n)-Z(n-1),DT ← DT+1

 9: If DX1≠0 OR DY1≠0 OR DZ1≠0 OR DT > 127 then GOTO 11 else GOTO 10

10: CNT ← 1  and GOTO 2

11: TDT ← DT-63, X(n-2) ← X(n-1), Y(n-2) ← Y(n-1), Z(n-2) ← Z(n-1)

12: X(n-1) ← X(n), Y(n-1) ← Y(n), Z(n-1) ← Z(n) DT ←0 and GOTO 2

13: DX2 ← X(n)-X(n-1),DY2 ← Y(n)-Y(n-1),DZ2 ← Z(n)-Z(n-1),DT ← DT+1

14: If DX2≠0 OR DY2≠0 OR DZ2≠0 OR DT > 127 then GOTO 16 else GOTO 15

15: CNT ← 2  and GOTO 2

16: DDX ← DX2-DX1, DDY ← DY2-DY1, DDZ ← DZ2-DZ1

17: If |DDX| > 63 then GOTO 18 else GOTO 21

18: DUMX(n)=.25X(n-1)+.75X(n)

    DUMX(n-1)=.25X(n-2)+.5X(n-1)+.25X(n)

    DUMX(n-2)=.75X(n-2)+.25X(n-1)

19: X(n) ← DUMX(n), X(n-1) ← DUMX(n-1), X(n-2) ← DUMX(n-2)

20: DX1=X(n-1)-X(n-2), DX2=X(n)-X(n-1)  and GOTO 16
```

Fig 11-a. Dower Collection and Decorrelation Algorithm.

```
21: If |DDY| > 63 then GOTO 22 else GOTO 25

22: DUMY(n)=.25Y(n-1)+.75Y(n)

    DUMY(n-1)=.25Y(n-2)+.5Y(n-1)+.25Y(n)

    DUMY(n-2)=.75Y(n-2)+.25Y(n-1)

23: Y(n) ← DUMY(n), Y(n-1) ← DUMY(n-1), Y(n-2) ← DUMY(n-2)

24: DY1=Y(n-1)-Y(n-2), DY2=Y(n)-Y(n-1)  and GOTO 16

25: If |DDZ| > 63 then GOTO 26 else GOTO 29

26: DUMZ(n)=.25Z(n-1)+.75Z(n)

    DUMZ(n-1)=.25Z(n-2)+.5Z(n-1)+.25Z(n)

    DUMZ(n-2)=.75Z(n-2)+.25Z(n-1)

27: Z(n) ← DUMZ(n), Z(n-1) ← DUMZ(n-1), Z(n-2) ← DUMZ(n-2)

28: DZ1=Z(n-1)-Z(n-2), DZ2=Z(n)-Z(n-1)  and GOTO 16

29: If FLAG=0 then GOTO 30 else GOTO 31

30: VLC ← (0:DX1,DY1,DZ1)

31: VLC ← (TDT:DDX,DDY,DDZ)

32: If MEMORY FULL then STOP else GOTO 33

33: X(n-2) ← X(n-1), Y(n-2) ← Y(n-1), Z(n-2) ← Z(n-1)

34: X(n-1) ← X(n), Y(n-1) ← Y(n), Z(n-1) ← Z(n)

35: TDT ← DT-63, FLAG=1 and GOTO 2
```

Fig 11-b. Power Collection and Decorrelation Algorithm.

Dower EKG Data Compressor

The Dower EKG compression technique, like the Tolan method, is a redundancy reduction procedure. The Dower compressor combines a zero order time compression operation with a second order difference reduction transformation to produce a decorrelated residual frame sequence. This residual frame sequence is then compressed by a specially tailored variable length code whose average codeword bit length approaches the entropy bound without the buffer overflow problem encountered with the ''optimal'' Huffman code.

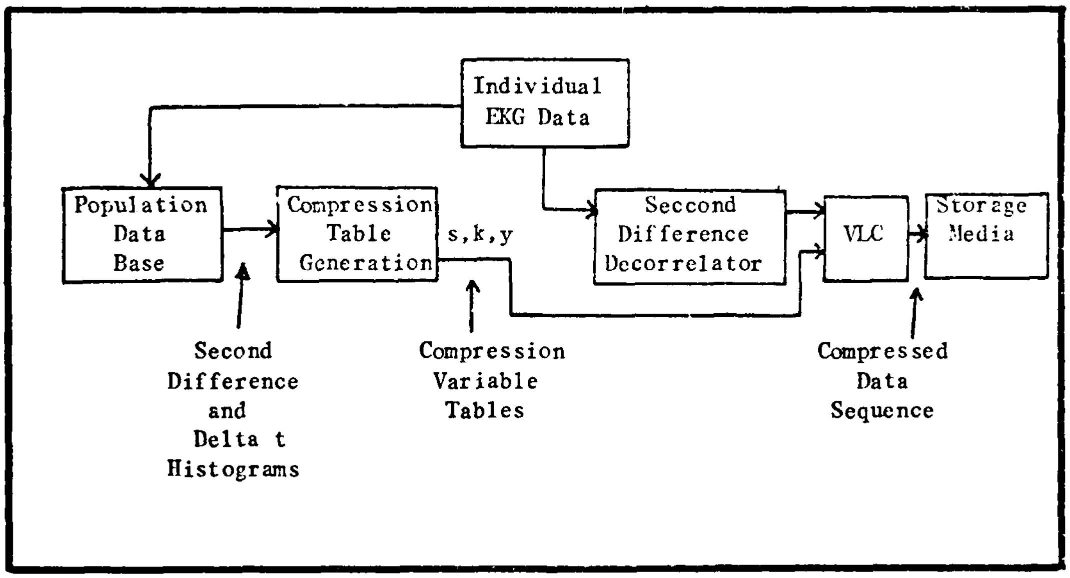


Fig 12. Dower EKG Data Compression System.

Dower Decorrelator. The Dower data collection and decorrelation operation (Figure 11) begins with a zero order time compression process where only those EKG sample values (x,y,z) which differ from the previous sample (Δx or Δy or $\Delta z \neq 0$) are saved along with a run length conter Δt. This operation produces a sequence of data frames (Δt: x,y,z). Following the creation of the data frames, a second difference with respect to frame number is performed generating a sequence of second order difference frames (Δt: $\Delta_f^2 x, \Delta_f^2 y, \Delta_f^2 z$).

The Dower algorithm operates on 8 bit data, hence there are 256 potential source symbols for each lead. When second differences are taken, the range increases to 1024 potential Δ^2 symbols. By experimental evidence, Dower and Berghofer (Ref 12) *have found that* $0 \pm 63 \Delta^2$ values are sufficient to reproduce all but the fastest EKG artifacts (e.g.,pacemaker spikes). The variable length encoder, therefore, is designed to expect 127 source symbols (values) and no more. To insure the Δ^2 dynamic range of 0 ± 63 is not exceeded, the Δ^2 values are limited by a preprocessor shown in steps 18-26 of Figure 11. This preprocessor is iterative, and irreversably modifies the three sample points which produced the $|\Delta^2| > 63$ until the Δ^2 value falls within the encoder range. With care taken to record the necessary initial conditions (first x,y,z and first $\Delta x, \Delta y, \Delta z$),this second difference frame sequence now contains all the significant information in the original sample sequence with

reduced interframe correlation.

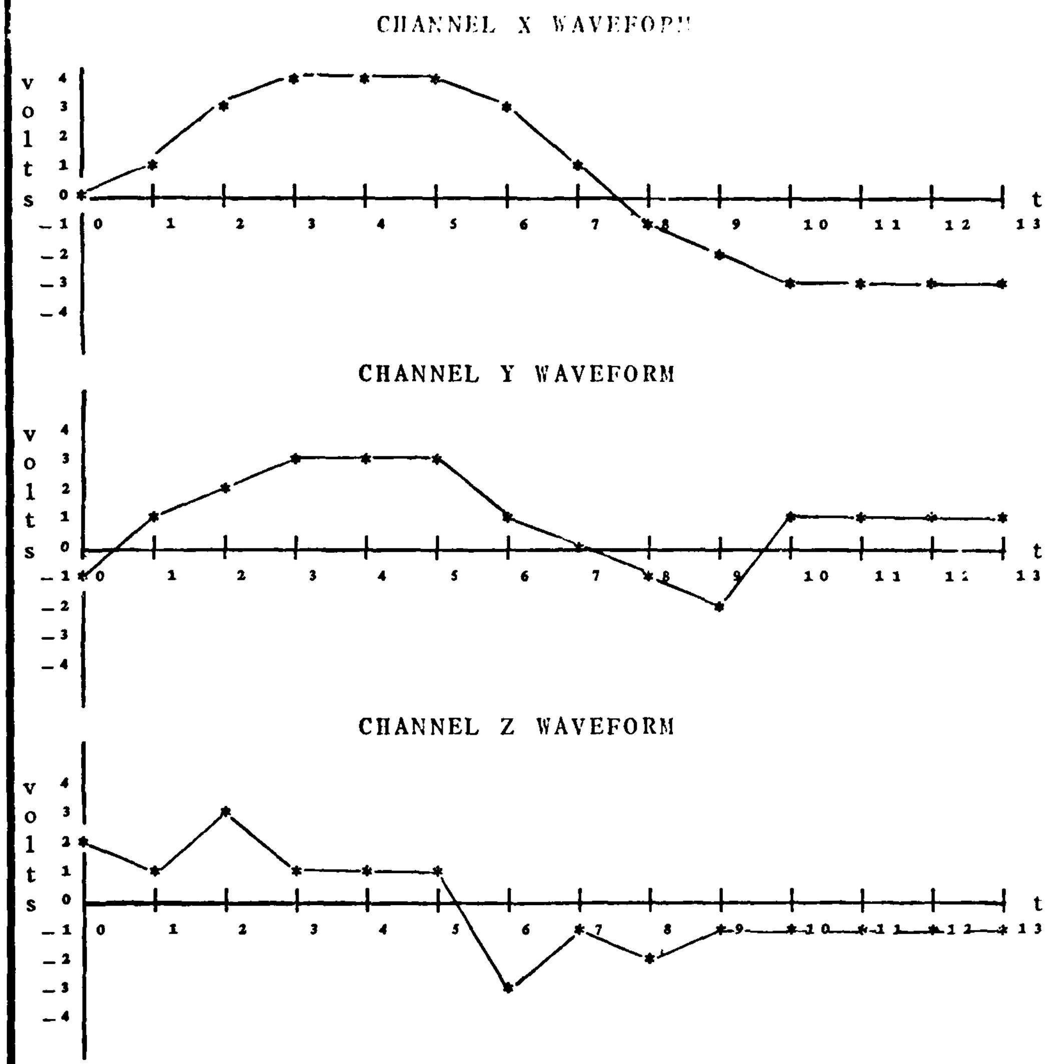
The mechanics of the Power decorrelator are best illustrated by an example. Figure 13 shows three hypothetical sample sequences. By applying the rules in the preceeding paragraph, a set of zero order data frames $(\Delta t{:}x,y,z)$ is formed. That is:

(0:0,-1,2),(0:1,1,1),(0:3,2,3),(0:4,3,1),(3:3,1,-3),(0:1,0-1),
(0:-1,-1,-2),(0:-2,-2,-1),(4:-3,1,-1)

To complete the correlation reduction process, the second difference with respect to frame number must now b performed. This results in the set of second differenc frames$(\Delta t{:}\ \Delta_f^2 x, \Delta_f^2 y, \Delta_f^2 z)$ shown below.

(0:0,-1,2),(0:1,2,-1) | (0:1,-1,3),(0:-1,0,0),(3:-2,-3,-2),
(0:-1,1,6),(0:0,0,-3),(0:1,0,2),(4:0,4,-1)

The data sets preceeding the verticle bar are the inital conditions necessary for reconstruction. The first data set is the first zero order time compression sample frame. The second frame is the first difference between the first two zero order data frames.



X=0 1 3 4 4 4 3 1 −1 −2 −3 −3 −3 −3

Y=−1 1 2 3 3 3 1 0 −1 1 1 1 1 1

Z= 2 1 3 1 1 1 −3 −1 −2 −1 −1 −1 −1 −1

Fig 13. Three hypothetical sample sequences.

Dower Entropy Encoder. The Dower entropy encoder also uses variable length codes for the second stage of the EEG data compressor. The Dower code, however, is significantly different than the prefix code used in the Tolan compression method.

As was mentioned previously, the Dower VLC is configured for 127 source symbols (8 bit Δ^2 values). Unlike the Tolan code, however, the Dower VLC does not perform a 1:1 mapping between a single source symbol and a single code symbol.

In the Dower coder, the source symbols (Δ^2_i) are mapped onto the state space of a 14 bit accumulator (A). This state space is partitioned into symbol regions R_i which are assigned according to the probability of occurrence of the source symbols. The size ($||R_i||$) of the symbol region R_i is given by the relation

$$||R_i|| = ds_i = s_{i+1}-s_i = Pr(\Delta^2_i)16384 \qquad (24)$$

where s_i is the initial state of R_i and s_{i+1} is the initial state of R_{i+1} . The initial state, s_i , is also determined by symbol probability. The symbol regions R_i corresponding to thos symbols which occur least often are assigned to the low end of the state space range such that if:

$$Pr(\Delta^2_k) > Pr(\Delta^2_n) > Pr(\Delta^2_m) > Pr(\Delta^2_j)$$

then

$$||R_k|| > ||R_n|| > ||R_m|| > ||R_j||$$

and

$$s_k > s_n > s_m > s_j$$

This relationship is illustrated in Figure 14. For those potential symbols Δ_i^2 which do no occur, 1 state is always assigned.

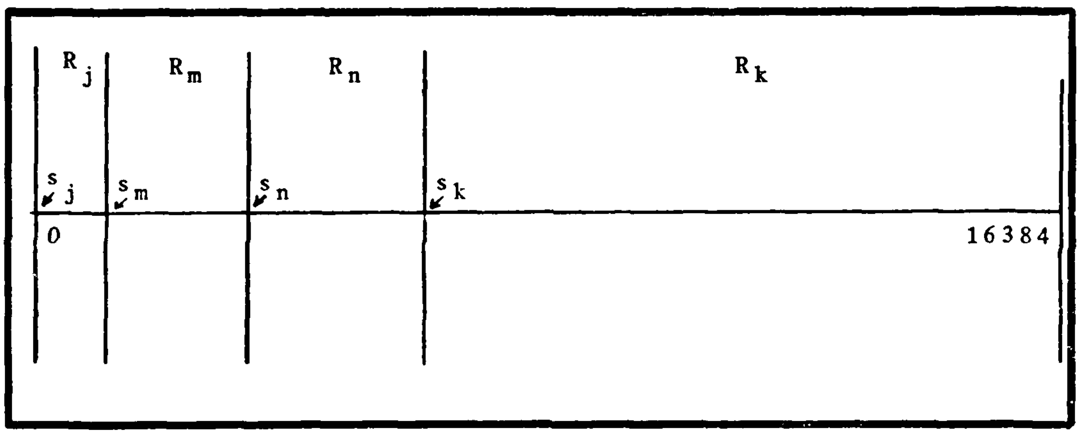


Fig 14. Example of State Space Partition for 4 Symbols.

The physical configuration of the Dower encoder is shown in Figure 15. As can be seen from Figure 15, the memory storage buffer for the encoded data is contiguous with accumulator and all accumulator bit shifts (right or left) also shift the entire memory buffer. Right shifting operations always shift a binary 0 into the most significant bit (MSB) of A.

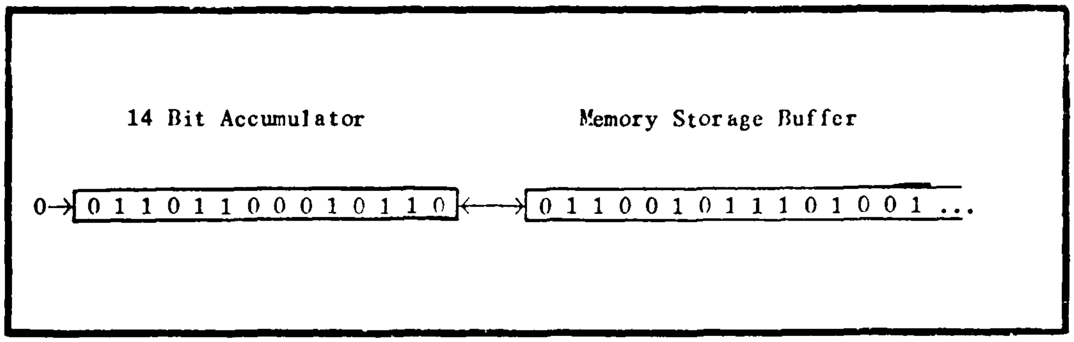


Fig 15. Dower Encoder Accumulator-Memory Buffer Interface.

A source symbol Δ_i^2 is encoded by adding the value s_i to the current contents of the accumulator. If the value in the accumulator is such that $s_i + A \geq 2^{14}$,then an <u>arithmetic overflow</u> condition would occur with a corresponding loss of data. To insure this overflow does not occur, the accumulator is first right shifted k_i times into memory. The value k_i is the <u>maximum number of right shifts</u> which would ever be necessary and can be found from the inequality:

$$2^{14} \leq 2^{k_i}(s_{i+1} - s_i) < 2^{15} . \tag{25}$$

The k_i values are tabulated in Table I for various values of $ds = s_{i+1} - s_i$. This right shifting <u>diminishes</u> the value in the accumulator (i.e., divides A by 2 per right shift) until the value in A can be safely mapped into a region R_i . The initial state, s_i can now be added to A without overflow occuring.

In some instances, the maximum shift k_i would not be

necessary (i.e., k_i -1 right shifts would still prevent overflow). To determine when this situation occurs, a third variable y_i is defined where

$$y_i = ds - 2^{14-k_i} \tag{26}$$

reperesents the <u>excess states</u> by which ds exceeds the next lowest integral power of 2.

Table I

Dower Code Variables Per Accumulator Partition Size

Partition Size	k	y
ds=1	14	0
1<ds<4	13	ds-2
4<ds<8	12	ds-4
8<ds<12	11	ds-8
16<ds<32	10	ds-16
32<ds<64	9	ds-32
64<ds<128	8	ds-64
128<ds<256	7	ds-128
256<ds<512	6	ds-256
512<ds<1024	5	ds-512
1024<ds<2048	4	ds-1024
2048<ds<4096	3	ds-2048
4096<ds<8192	2	ds-4096
8192<ds<16384	1	ds-8192

If the value in A < y_i after the initial k_i shifts, then one too many right shifts occured. A is then <u>left shifted</u> once and s_i added to A without fear of overflow. If A $\geq$ y_i then a full k_i right shifts were required. To insure maximum efficiency in the next encoding, however, the position of the accumulator in state space map should be

right justified within the current symbol region. This right justification is obtained by adding $y_i + s_i$ to A whenever $A \geq y_i$.

By use of the variable y_i maximum efficiency in the number of right shifts (i.e. information bits stored to memory) can be obtained. Dower and Berghofer (Ref 12) assert that, on the average, the number of right shifts per source symbol (Δ_i^2) will approach quite closely to the value $-\log_2 Pr(\Delta_i^2)$ which is the self information ''content'' of the Δ_i^2 value (see appendix B).

The algorithm which implements the Dower encoder is shown in Figure 15. This algorithm assumes that the variable tables for $\Delta_f^2 x, \Delta_f^2 y, \Delta_f^2 z$ and Δt have been calculated from the second difference histograms.

The Dower decoding operation is just the inverse of the encoding process. For decoding, the current state of the accumulator is mapped into the state space table of the current frame variable($\Delta t, \Delta_f^2 x, \Delta_f^2 y, \Delta_f^2 z$). Once the correct region R_i, is determined, then s_i is subtracted from A. If A $< 2y_i$ then one right shift is performed followed by k_i left shifts. If $A \geq 2y_i$, then y_i is subtracted from A followed by k_i left shifts. For a more detailed example of the Dower encoding/decoding operation, the reader is referred to Dower and Berghofer (Ref 12).

Although conceptually more difficult than the Tolan VLC, the algorithmic structure of the Dower entropy encoder is only slightly more complex. As was shown above, the

Dower VLC can be implemented solely with shifting and table look up operations. The Dower method does, however, require that <u>code tables</u> (s_i, k_i, y_i) be constructed prior to the encoding process.

1: Entry $(\Delta t, \Delta_f^2 X, \Delta_f^2 Y, \Delta_f^2 Z)$

2: IF this is the first frame stored then Acc$\leftarrow$0 else GOTO 3

3: DDVAR $\leftarrow \Delta t$, k$\leftarrow$ k(Δt), s$\leftarrow$ s(Δt), y$\leftarrow$ (Δt)

4: RIGHT SHIFT Acc k times

5: IF Acc $<$ y then GOTO 8 else GOTO 6

6: LEFT SHIFT Acc once

7: Acc $\leftarrow$ Acc + s and GOTO 9

8: Acc $\leftarrow$ Acc + s + y

9: IF memory is full, then SET eom flag and RETURN else GOTO 10

10: IF DDVAR=Δt then GOTO 11 else GOTO 12

11: DDVAR $\leftarrow \Delta_f^2 X$,k $\leftarrow$ k$(\Delta_f^2 X)$,s $\leftarrow$ s$(\Delta_f^2 X)$,y $\leftarrow$ y$(\Delta_f^2 X$) and GOTO 4

12: IF DDVAR=$\Delta_f^2 X$ then GOTO 13 else GOTO 14

13: DDVAR $\leftarrow \Delta_f^2 y$,k $\leftarrow$ k$(\Delta_f^2 Y)$,s $\leftarrow$ s$(\Delta_f^2 Y)$,y $\leftarrow$ y$(\Delta_f^2 Y$) and GOTO 4

14: IF DDVAR=$\Delta_f^2 Y$ then GOTO 15 else RETURN

15: DDVAR $\leftarrow \Delta_f^2 Z$,k $\leftarrow$ k$(\Delta_f^2 Z)$,s $\leftarrow$ s$(\Delta_f^2 Z)$,y $\leftarrow$ y$(\Delta_f^2 Z$) and GOTO 4

Fig 16. Dower VLC algorithm.

Tolan versus Dower

The similarites and differences between the Tolan and Dower EKG compression techniques are now discussed. The decorrelators are compared first.

Both the Dower and the Tolan decorrelators use a sceond difference technique where time compression is used to eliminate the storing of the most common Δ^2 value of zero. The Tolan algorithm, however, continuously calculates the second difference and applies these differences to the Tolan VLC. If a long ''run'' of $\Delta^2 = 0$ values occur, the Tolan encoder overflows at $\Delta t = 128$. This run counter overflow forces a storage ''dump'' to the VLC with a resultant loss in efficiency.

The Dower decorrelator approaches time compression slightly differently. In the Dower algorithm, only those sample points where the EKG data was changing (i.e., Δx.or.Δy.or.$\Delta z \neq 0$) are saved forming data frames. A second difference with respect to frame number is performed and the Δ_f^2's and Δt are fed to the VLC. Although this frame methodology appears more efficient than the Tolan technique, the Dower decorrelator is still constrained by a maximum Δt of 127 (i.e., maximum Dower VLC code range). On the basis of this analysis, is appears that both the Tolan and the Dower decorrelators have similar performance.

Since the Tolan and Dower decorrelators appear about equally efficient, the real compression payoff is in the variable length encoders. In the Tolan VLC, code words as

long as 1024+2 bits are feasible. These long codewords wo[illegible] only occur if an extremely large and fast ''spike'' (e.g. a pacemaker pulse) appeared in the data. In the population as a whole, spikes of this magnitude occur very infrequently. Nevertheless, the Tolan compression efficiency will degrade seriously in an enviroment where ''impulsive'' artifacts appear.

The Dower VLC is designed to minimize sensitivity to impulses. In the Dower system, the VLC is configured such that the <u>longest</u> <u>codeword</u> (i.e., the number of right shifts to memory) is 14 bits. Extremely large signal spikes are numerically filtered to reduce their second difference within the 0 ± 63 range. Such drastic limiting action would occur infrequently, however, and not seriously affect the reproduction of the EKG.

As an example of their performance, let a second difference of 45 be encountered by both the Tolan and Dower encoders. The Tolan VLC would require 47 bits to encode this data. The exact codeword size of the Dower routine is dependent on the $\Delta^2 = 45$ probability of occurrence. Nonetheless, the ''codeword'' is always $\leq$ 14 bits long ; an obvious increase in efficiency over 47 bits.

A calculation of the performance differential between the Tolan and Dower routines will be estimated in chapter 5. For now it is sufficient to say that the Dower EKG compressor outperforms the Tolan method. A short synopsis will now be made of two other EKG compression techniques.

Other Microcomputer EKG Compressors.

Two other EKG compression techniques were discovered by this author. The first is referred to as the Turning Point Method and is discussed in reference 32. The second method is currently in use by Marquette Electronics for data compression in a commercial EKG ''cart''. The Turning Point technique is presented first.

Turning Point Algorithm. The turning point algorithm is, by definition, a 2:1 data compressor where one of two consecqutive sample points is discarded. The algorithm which determines which sample point is discarded is as follows. The first sample point is stored and assigned as the reference point (X_0). The next two consecqutive points become X_1 and X_2 . With 3 sample points there are 8 ''paterns'' or combinations which reflect the ''trends'' in data (see Figure 17). The Turning Point algorithm stores the circled point (Figure 17) which becomes the new reference point X_0 . The point not circled (X_1 or X_2) is discarded. The next two points are sampled, their values are assigned to X_1 and X_2 , and the process repeated.

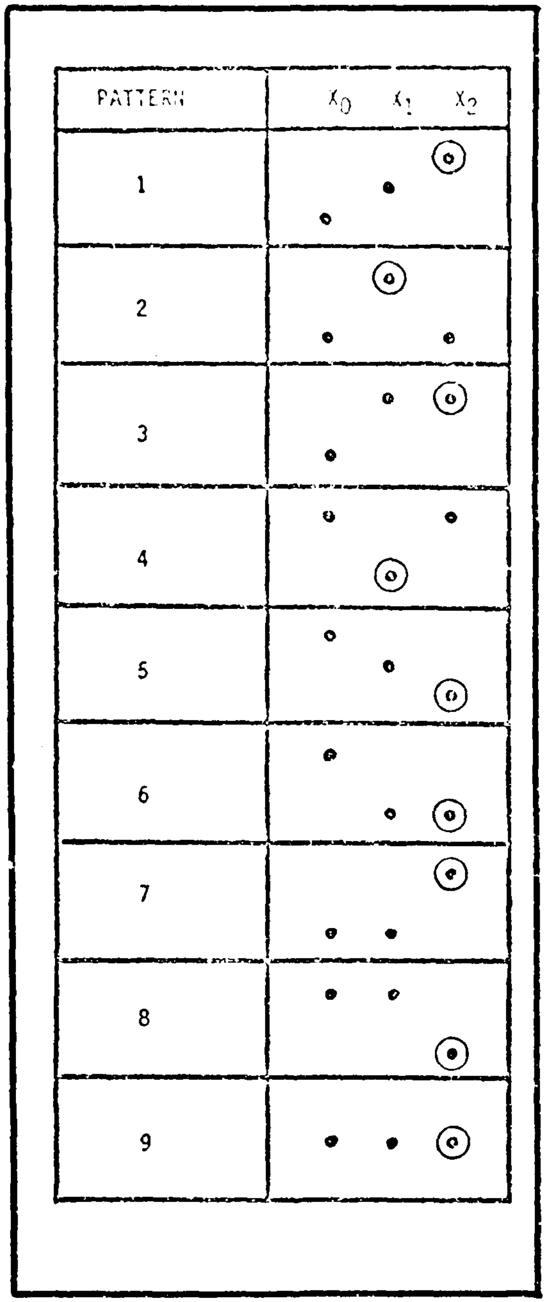


Fig 17. Turning Point Patterns (from Ref 32:6.61).

It can be shown (Ref 32:6.59-6.65) that :

if $(X_2 - X_1)*(X_1 - X_0) < 0 \quad X_0 = X_1$

if $(X_2 - X_1)*(X_1 - X_0) \geq 0 \quad X_0 = X_2$

hence the sign of the product of consecqutive first differences determines the ''signigicant'' point to be saved.

The advantage of the Turning Point compressor is speed of execution. Execution of this algorithm on even the slowest microprocessor would pose no problem at all. The disadvantages of the turning point routine are numerous, however. First the routine discards data so that reconstruction of the original sample sequence is not possible. In this regard, the Turning Point algorithm falls in the class of Entropy Reducing (ER) techniques as was discussed in chapter 2. Second no attempt is made to use the probabilistic distribution of the EKG to enhance data compression as is done with the Tolan and Dower routines. This obviously leads to inefficiency. Finally, this routine only produces a 2:1 compression ratio; incredibly poor in relation to the other techniques already discussed in this thesis. Only where the simplest technique is necessary, would the Turning Point technique be beneficial.

Marquette Algorithm. The Marquette algorithm was developed by Marquette Electronics, Milwaukee, Wisconsin. This similar to the Dower and Tolan algorithms in that variable length encoding is used to compress the output of a

''difference'' decorrelator.

In contrast to the Dower and Tolan techniques, the Marquette compressor only calculates the first difference for input into the VLC. The Marquette variable length encoder stores the first difference data as 1,3,5,or 7 <u>nibbles</u> where a nibble is defined as 4 bits. These nibble codes are arranged as follows:

<u>Range of Difference</u>	<u>Code Length in Nibbles</u>
(-7,7)	1
(-127,127)	3
(-2047,2047)	5
(-32767,32767)	7

To encode the first differences, and delineate between code words, the following rules apply:

1) Differences must be coded on the smallest possible range, and attempts to encode a difference of +5, for example, using more than 1 nibble will result in a decoding error.

2) Single nibble codes are difference plus 8. A nibble value of zero does not occur.

3) Three nibble codes start with a single zero nibble. The remaining two nibbles are obtained as follows:
   (1) Positive Differences +8
   (2) Negative Differences +7

4) Five nibble codes start with two zero nibbles. The remaining three nibbles are obtained as follows:
   (1) Positive Differences +128
   (2) Negative Differences +127

5) Seven nibble codes start with three zero nibbles. The remaining four nibbles are obtained as follows:

(1) Positive Differences +2048
(2) Negative Differences +2047

6) The sequence of nibbles in a code starts with the zero flag nibbles (if any) followed by the most significant through least significant nibble.

The above set of encoding rules were obtained from Mr. Tom Divers, Marquette Electronics project engineer (Ref 11).

From their own analysis, Marquette has shown (Ref 11) that at a 250 Hz sampling rate, 89.1 percent of the first differences fall within the $\pm$ 7 range with 99.8 percent falling within a $\pm$ 127 range. Marquette reports that at an A/D precision of 10 bits, an average of 4.89 bits/sample (across the total EKG population) is obtained with their compression routine.

The Marquette EKG compression appears to work well, even with a first order difference correlation reducer. The Marquette variable length encoder , however, is tailored to the 8 bit ASCII data communications enviroment and is not ''optimum'' in any sense. The Marquett VLC does perform ''exact'' redundancy reduction entropy compression with sufficient efficiency to make this encoding scheme commercially viable.

<u>Chapter III Summary</u>.

This chapter has looked in detail at two EKG data compression techniques which perform ''exact '' redundancy reduction. The Tolan routine, which was implemented by this author (see chapters 4 and 5) decorrelated the sample

sequence data by performing a second order time compression operation. The residual sequence resulting from the decorrelator was ''compressed'' by a uniquely decodable variable length code which was shown to be ''suboptimal'' with respect to the ''optimal'' Huffman code.

The Dower compression technique also performed a second difference operation, but preceeded the second order ''differencer'' by a zero order time compressor. The Dower zero order time compressor produced a sequence of ''data frames'' which in turn were converted to a sequence of second order difference frames with respect to frame number. Since the Dower VLC limited the Δt time compression counter to a maximum of 127 (as did the Tolan VLC), input symbols, the Dower and Tolan decorrelators were considered to perform equally well. The Dower entropy encoder was also shown to be a variable length coding operation but not a ''prefix'' code as was the Tolan VLC. The Dower VLC maps decorrelator ''symbols'' into the ''state space'' of 14 bit accumulator which encodes data by adding the initial address of a symbol's state space region to the accumulator. To prevent accumulator overflow, the accumulator data is shifted out to a memory storage buffer. The ''number of shifts'' necessary to prevent accumulator overflow represent the codeword size and it was shown that the Dower VLC approached the ''entropy'' bound of the decorrelated input sequence.

The chapter was concluded by a synopsis of two other

EKG compression techniques. The first of these two, the Turning Point algorithm, was shown to be of marginal use because it is not an ''exact'' technique and produces compression of only 2:1. The second technique, however, was the Marquette compression system and it was shown to produce acceptable compression worthly of commercial application.

From the available algorithms, the Dower compression technique has the capacity to produce the best ''exact'' compression of any of the techniques studied in this chapter. An interesting experiment would be the combining of the second order interpolator (discussed in chapter 2) used by Ruttiman and Pipberger (Ref 28) and the variable length encoder used by Dower (Ref 12). This combination should prove to be very powerful and effective and is left for futher study.

The next chapter in this thesis discusses the configuration of the <u>EKG Data Acquistion and Analysis System</u> assembled by this author to test the Tolan EKG compression algorithm.

IV. EKG-Data Acquisition and Analysis System

The EKG-Data Acquisition and Analysis System (EKG-DAAS) was assembled for this thesis as the testbed on which experimental EKG data could be acquired,compressed, analyzed,stored, and reconstructed. The EKG-DAAS design can be separated into the categories of hardware and software. The hardware is discussed first.

EKG-DAAS Hardware.

The EKG-DAAS was constructed around the Motorola Exorciser microcomputer (appendix E) which uses a 6800 microprocessor for its central processing unit (CPU). In the EKG-DAAS, the Exorciser is configured with 32 kilobytes (K) of read/write (RAM) memory and 16 K of read only memory (ROM). In addition, the Exorciser was equiped with the EXBUG debuging module which allowed interactive program debugging with preselectable software breakpoints, execution tracing modes, and CPU register display.

To provide extended memory, a Midwest Scientific Instruments (MSI) FD-8 Disk Memory unit was interfaced to the Exorciser and provides approximately 290 K of online user memory. The FD-8 is accessed by a MSI Disk Operating System (DOS) and communicates with the CPU via a MEX6820 Input/Output Module installed in the Exorciser chassis.

Terminal input/output (I/O) is accomplished by means of a standard RS-232 serial interface which has switch selectable baud rate from 110 to 9600 baud. In the EKG-DAAS configuration, the Exorciser serial ''port'' was connected in parallel with a Heathkit H-14 dot matrix line printer for program listings and data printouts.

Data I/O was accomplished by means of a Sinetrac ST-6800 Analog/Digital-Digital/Analog (A/D-D-A) converter module which samples and digitizes analog data to 12 bit precision. The ST-6800 has the capability of sampling 32 distinct analog channels (A/D) as well as output 2 channels (D/A) with simple memory addressed LDA (load) and STA (store) instructions. For the EKG-DAAS, the ST-6800 was setup for ± 5 volt, 2's compliment data and was addressed (A/D ch 0) at E400 Hexadecimal (Hex). The internal configuration of the Exorciser as used in the EKG-DAAS is illustrated in Figure 18.

To uniformly sample the EKG input data via the ST-6800 required the use of an external <u>interrupt timer</u> as is shown in Figure 19. This timer allowed data sampling rates between 300 and 700 Hertz but for the duration of this research was set, and calibrated, at 500 Hertz. The interrupt was interfaced to the Exorciser via an interrupt line on the ST-6800.

Considerable problems arose in this thesis due to hardware problems associated with the FD-8 Disk Memory. The original configuration of the EKG-DAAS used two FD-8 systems

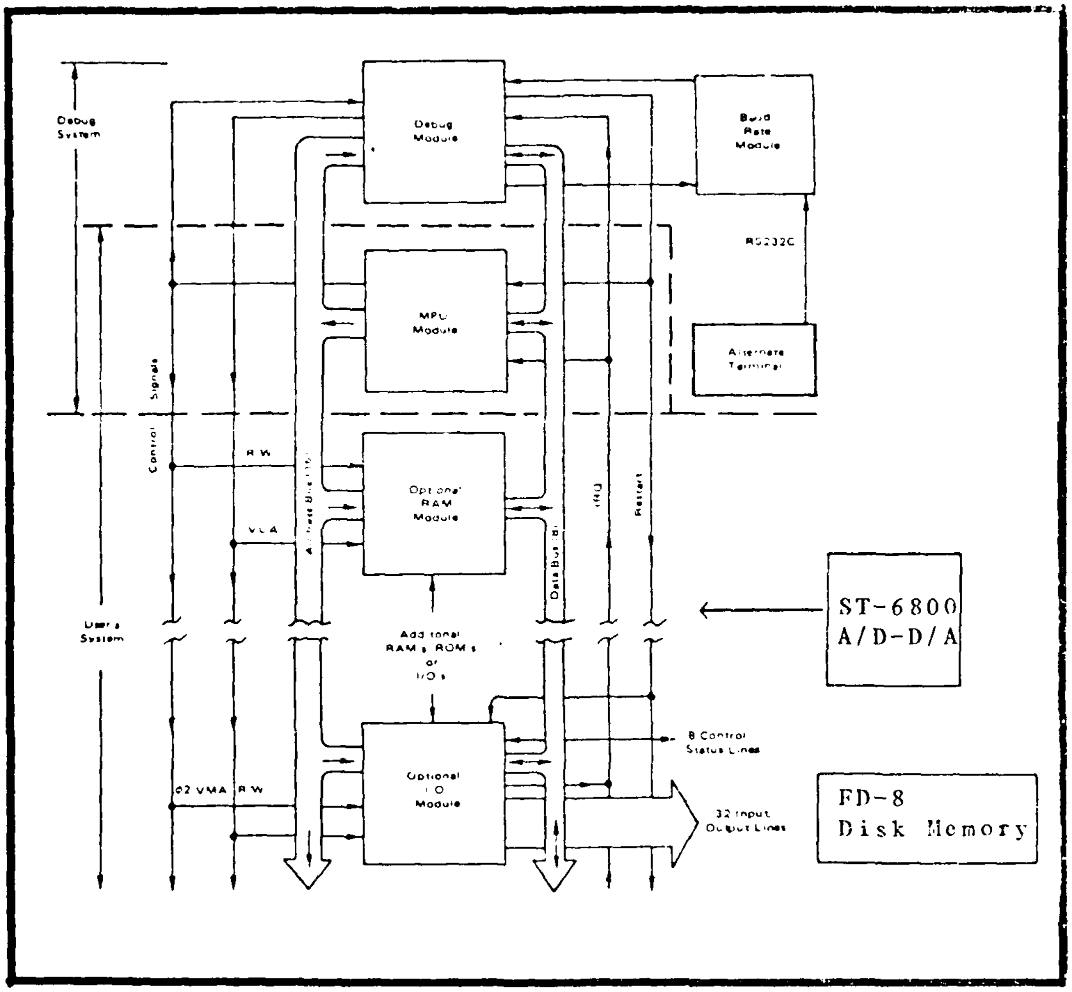


Fig 18. EXORCISER Component Module Layout (From Ref 22).

but one failed about midway through the software development. This failure caused a major rewrite of the thesis software and destroyed several <u>weeks</u> of work. Although the EKG-DAAS can now operate in a one disk environment, considerable ''manhandling'' of the data diskettes is necessary.

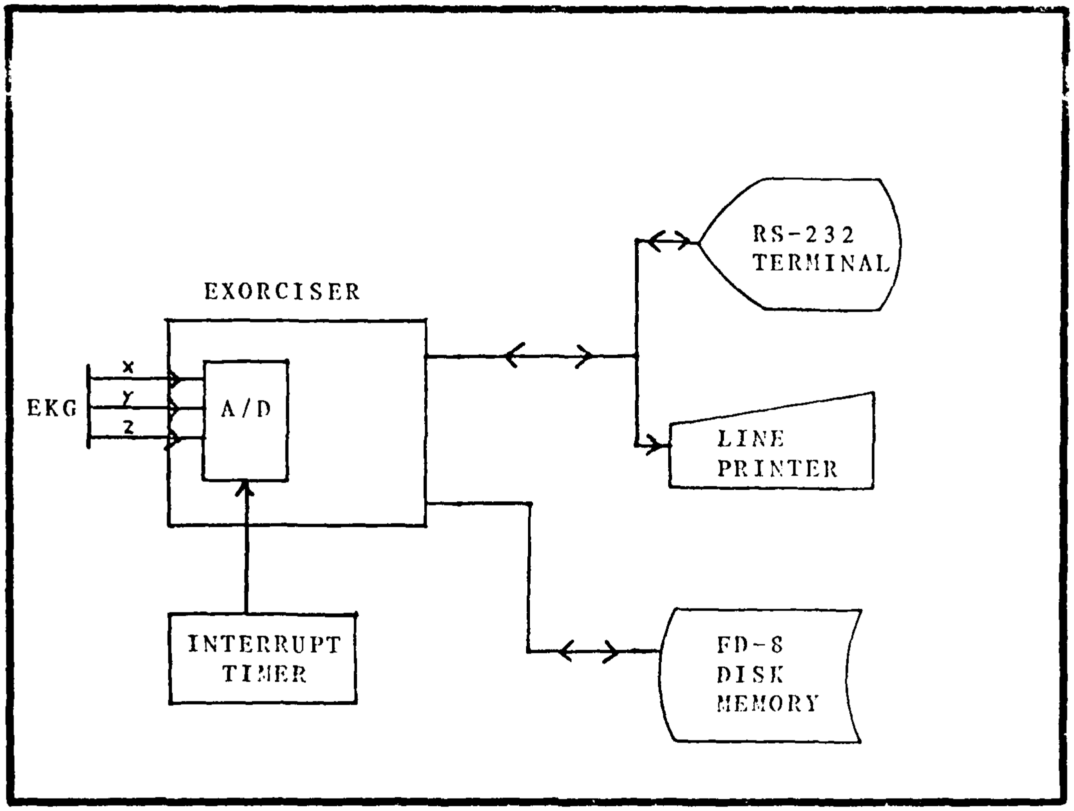


Fig 19. EKG-DAAS Hardware Configuration

Further information concerning the Exorciser's hardware capability can be obtained in appendix E and reference 22. Attention is now turned to the EKG-DAAS software.

EKG-DAAS Software.

The EKG-DAAS software was written in 6800 assembly language and controls all aspects of terminal, disk memory, and A/D-D/A operation. The EKG-DAAS programs consist of approximately 4300 lines of assembly language and are listed in appendix C.

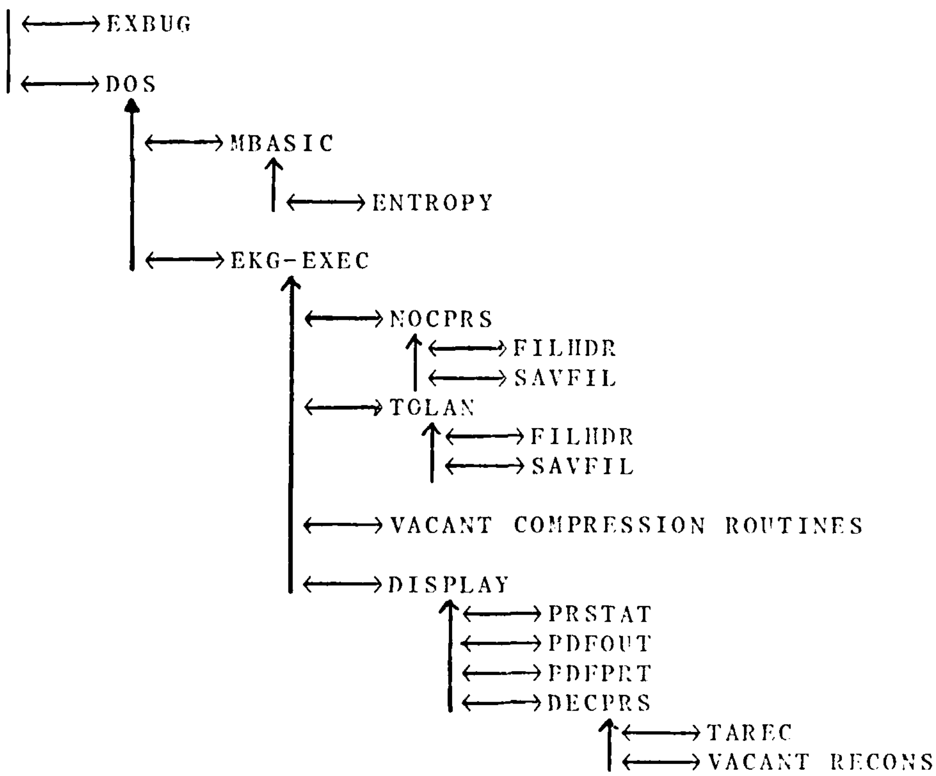
The software used in the EKG-DAAS was written in 6800

assembly language for two reasons: 1) no _high_ order language (e.g. PASCAL,FORTRAN) was available on the Exerciser and ; 2) speed limitations imposed by the thesis requirement for _online_, _real time_ EKG data compression made it imperative that the compression programs run as fast as possible. The laborious task of writing and testing assembly language slowed software development to the point where only one EKG data compression/reconstruction routine (Tolan) was completed.

The EKG-DAAS software is integrally tied to the MSI DOS (Ref 23). All disk I/O operations initiated by the EKG-DAAS routines flow through the MSI-DOS and hence the DOS must be ''live'' somewhere in memory. To insure that the DOS routines are always available, the DOS was disassembled and relocated in high memory ROM (C400 Hex).

The basic flow of EKG-DAAS program control is illustrated in Figure 20. The EKG-DAAS software is broken into _overlayed_ modules which are called into memory and executed by EKG-EXEC and DISPLAY. The basic memory map and overlay structure is illustrated in Figure 21.

As can be seen in Figure 21, extensive _memory management_ was required in order to allow a sufficiently large memory buffer for the EKG data. As configured in Figure 21, the EKG-DAAS could collect 11.6 seconds of _uncompressed_ (3 leads,8 bits/lead,500 samples/sec) EKG data. For compression with the Tolan algorithm, a maximum of 26.2 seconds of data (TA1359PA, appendix D) was



DOS.......Disk Operating System
EXBUG.....Debug Monitor Program
MBASIC....Modified MSI BASIC Interpreter
EKG-EXEC..EKG-DAAS Executive Program
NOCPRS....Data Collection Without Compression
TOLAN.....Data Collection With Compression (Tolan algor)
DISPLAY...Data Output Module
PRSTAT....Statistic Printout Driver
PDFPRT....Amplitude Histogram Printout Driver
PDFOUT....Amplitude Histogram Oscilliscope Display Driver
DECPRS....Reconstruction and Display Module
TAREC.....TOLAN Reconstruction and Display (Oscilliscope)

Fig 20 EKG-DAAS Software Control Flowgraph.

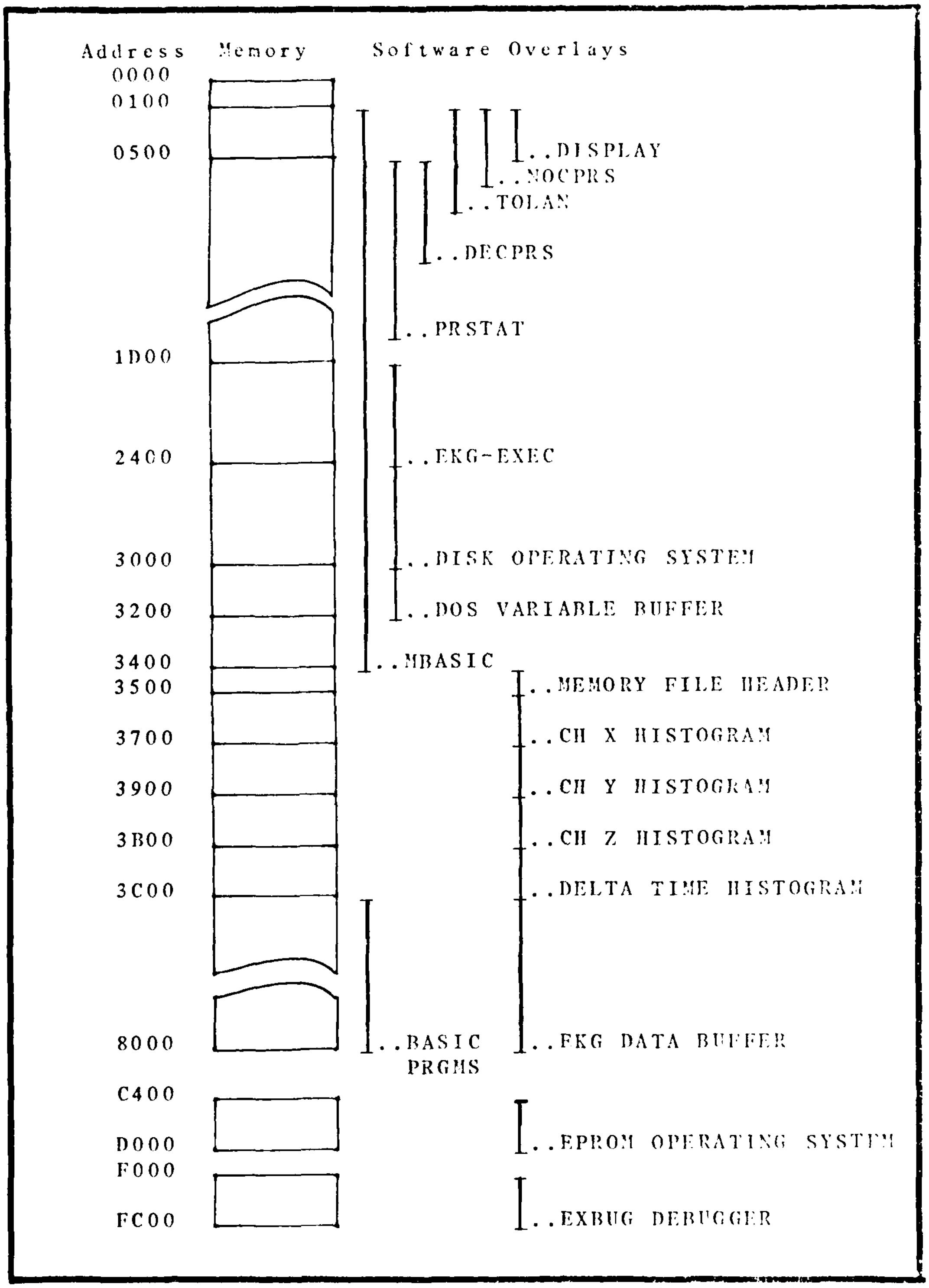


Fig 21. EKG-DAAS Overlay Structure and Memory Map.

collected.

With the basic structure of the EKG-DAAS software defined, this chapter will now examine the components of the EKG-DAAS software in more detail. In the next section of this chapter, a software module will be described along with a simplified flowchart of that module's operation.

EKG-EXEC. EKG-EXEC is the exectuive command module which controls the execution flow of the EKG-DAAS system. Upon input of a command number, EKG-EXEC loads the appropriate overlay routine into the program work buffer (0100-1D00 Hex) and then passes control to that overlay. The above command and control operation is illustrated in Figure 22.

In addition to the command ''handler'', EKG-EXEC contains the utility subroutines FILHDR, SAVFIL, HXASC, OVRLAY, and PDFPRT. These subroutines are described as follows:

> FILHDR. FILHDR clears the memory data buffer, initializes the statistics buffer variables, and queries the console for data such as FILENAME,SUBJECT,DATE,etc.
>
> SAVFIL. SAVFIL reads the filename in the memory buffer header and then writes the memory file to disk. Disk I/O is passed through EOS subroutines.
>
> HXASC. This subroutine converts hexadecimal data to ASCII for display on the terminal and printer devices.
>
> OVRLAY. OVRLAY is the routine which actually performs the overlay function. After an overlay is loaded into memory, OVRLAY jumps program control to the overlay program.
>
> PDFPRT. PDFPRT prints the amplitude distribution to the terminal device (printer). Although resident in EKG-EXEC, PDFPRT is called only by the DISPLAY module.

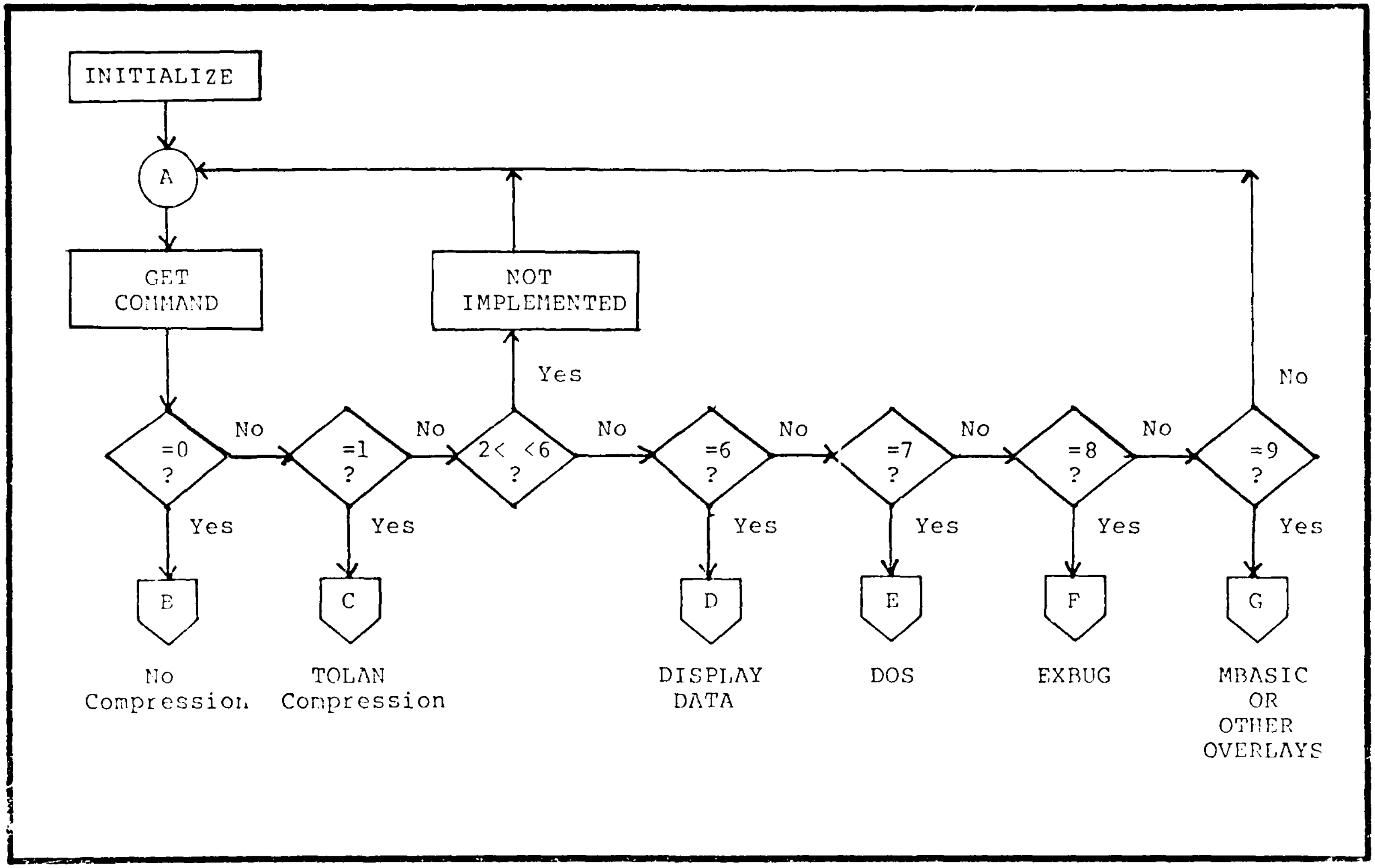


Fig 22. EKG-EXEC Functional Flowchart

NOCPRS. NOCPRS is a data acquisition module in which the EKG waveform is sampled and stored without data compression. The data is rounded to 8 bits from 12 bits. The reason for this rounding is explained in chapter 5.

This module was constructed for two reasons : 1) uncompressed data was considered useful for doing experimental studies on potential data compression techniques implemented after the original data collection session and ; 2) this module was the structure around which the Tolan (and potentially other) compression routines were built. The basic operation of NOCPRS is illustrated in Figure 23.

As is seen in Figure 23, NOCPRS does more than just sample the EKG. Statistical parameters are collected and updated during an EKG data collection. These parameters are used to measure the real time performance of the compression (no compression) software. A detailed description of these measurement parameters is described in chapter 5.

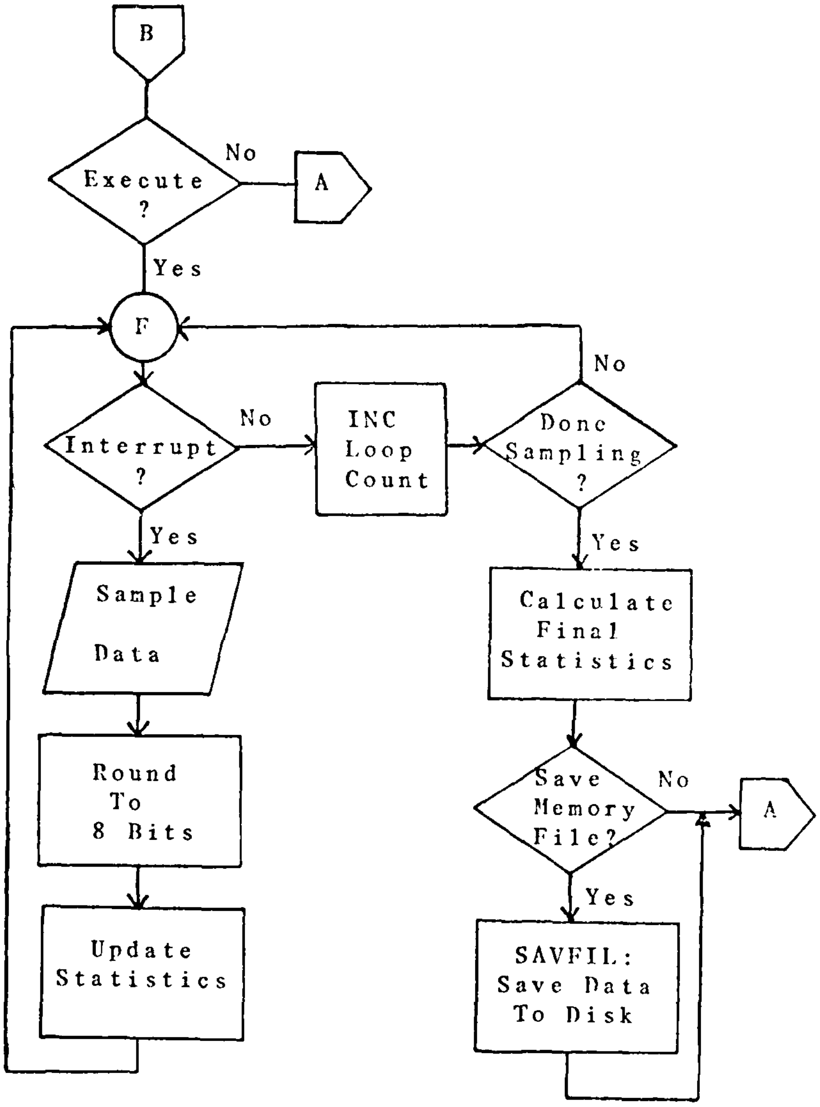


Fig 23. NOCPRS Functional Flowchart.

DISPLAY. DISPLAY is a command module similar in construction to EKG-EXEC. DISPLAY controls the data output modules which display the sampled EKG data and data statistics to two output devices. The first device is the terminal (printer) and the second device is an oscilloscope. The basic command structure of DISPLAY is shown in Figure 24. DISPLAY is broken into 5 working submodules. These modules are described as follows:

PRSTAT. This submodule reads the memory file header (3C00-3D00 Hex), formats the statistical data found there, and prints this data to the terminal (printer). The statiscal data in appendix D was generated by PRSTAT.

PDFPRT. This submodule prints the memory file histogram tables to the terminal (printer). PDFPRT output is also listed in appendix D.

PDFOUT. PDFOUT scans a user selected lead histogram (X,Y,Z) and formats the data for display to an oscilloscope. The data is output via D/A ch 0. An example of PDFOUT output is found in Figure 31 in chapter 5.

DECPRS. The DECPRS module scans the memory file header and identifies the compression technique which was used to encode the data in memory. The appropriate decompression algorithm is then selected and the data decoded and output on D/A channel 0.

LOAD. LOAD initiates a data file load from disk memory to RAM. This load is performed by EOS routines called by load.

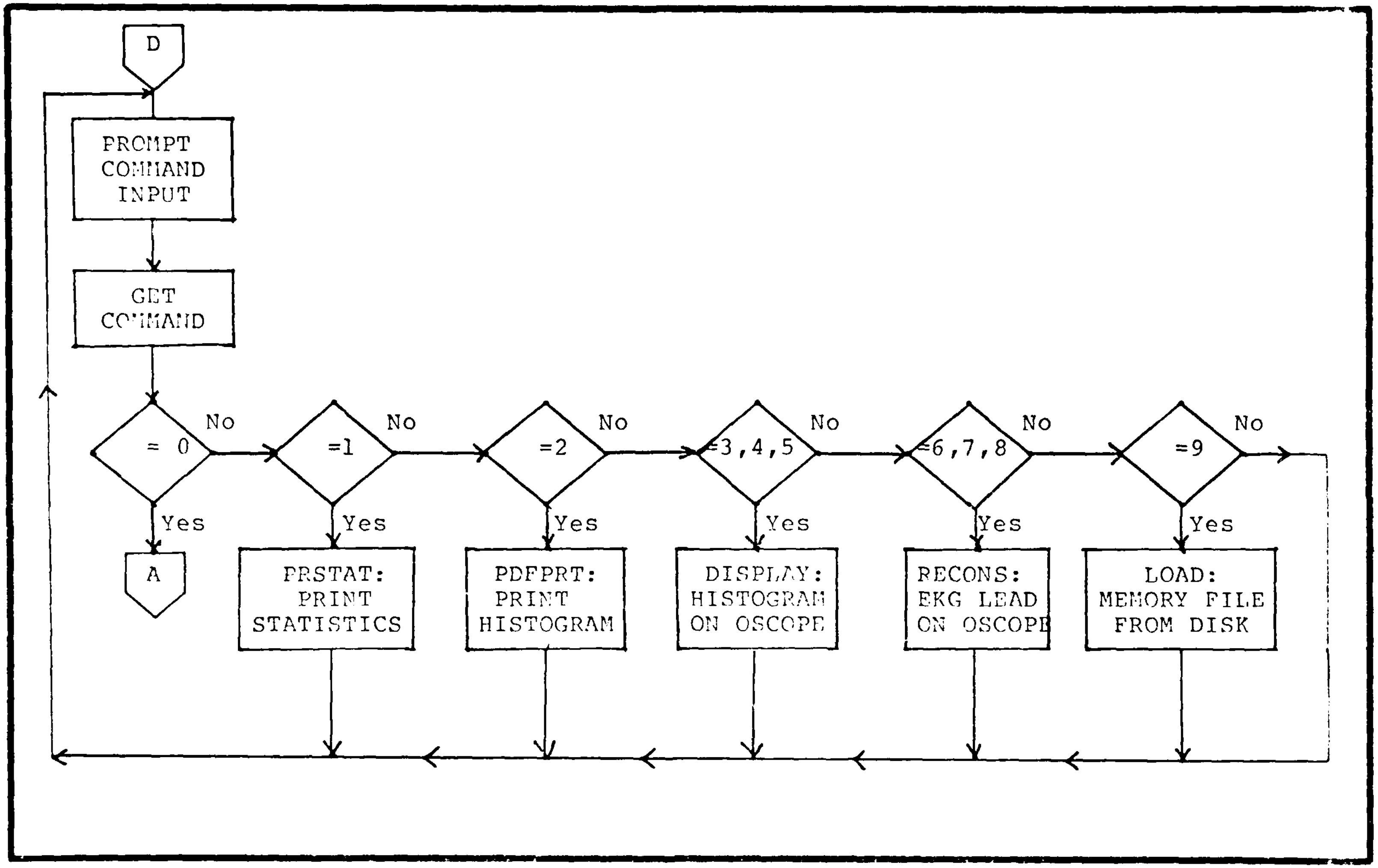


Fig 24. DISPLAY Functional Flowchart.

TOLAN. The TOLAN module is the ''heart'' of the EKG-DAAS and performs the actual data compression on the sampled EKG. As was described in chapter 3, the TOLAN algorithm first performs a second difference time compression operation followed by a variable length encoder. The operation of the TOLAN compression module is shown in Figure 25.

To detect sample clock (or CPU clock) drift, time calibration operations are performed prior to and after the data collection run. Other statistical data parameters are also collected allowing post collection measurement of compression performance.

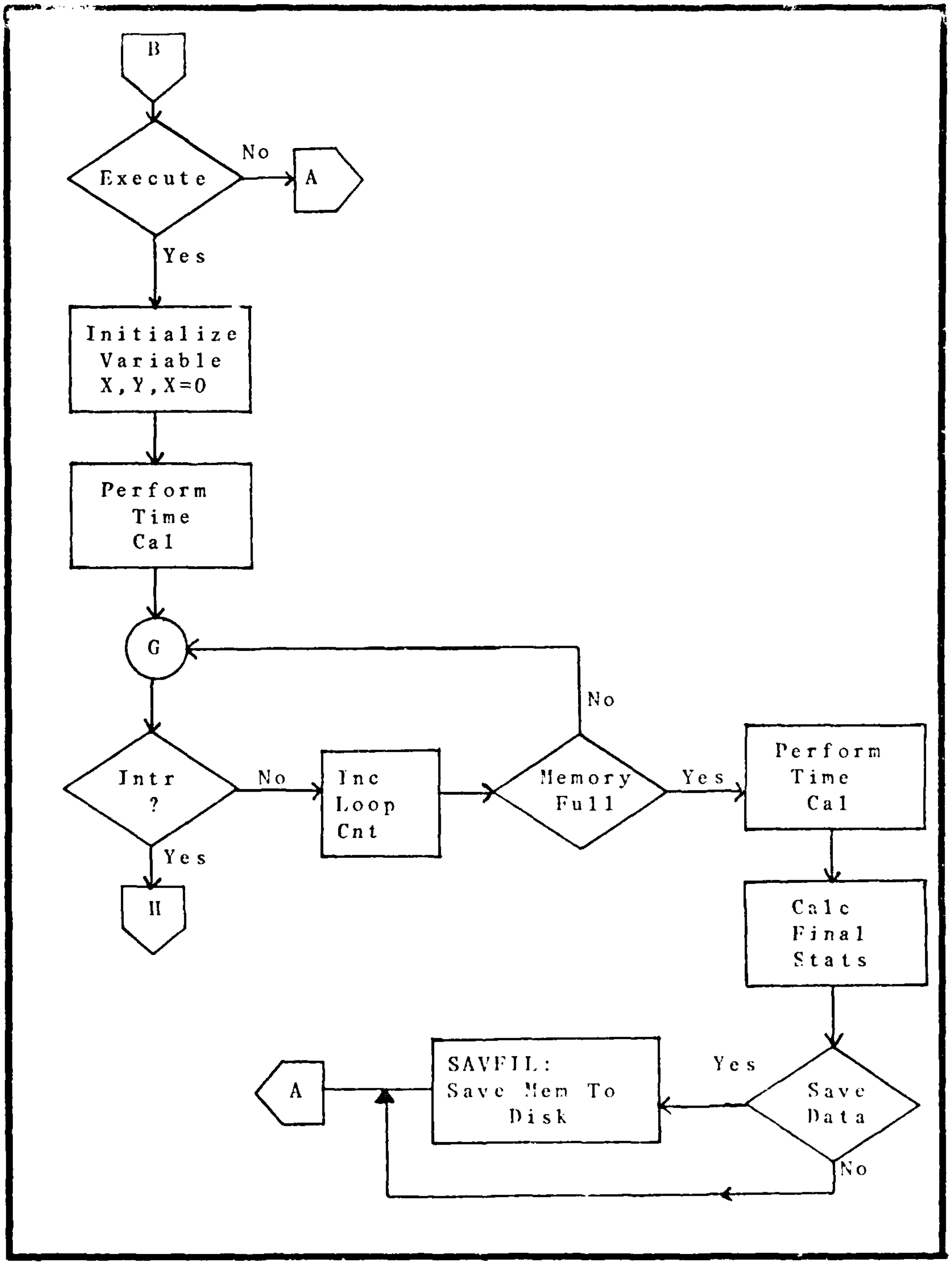


Fig 25a. TOLAN Functional Flowchart.

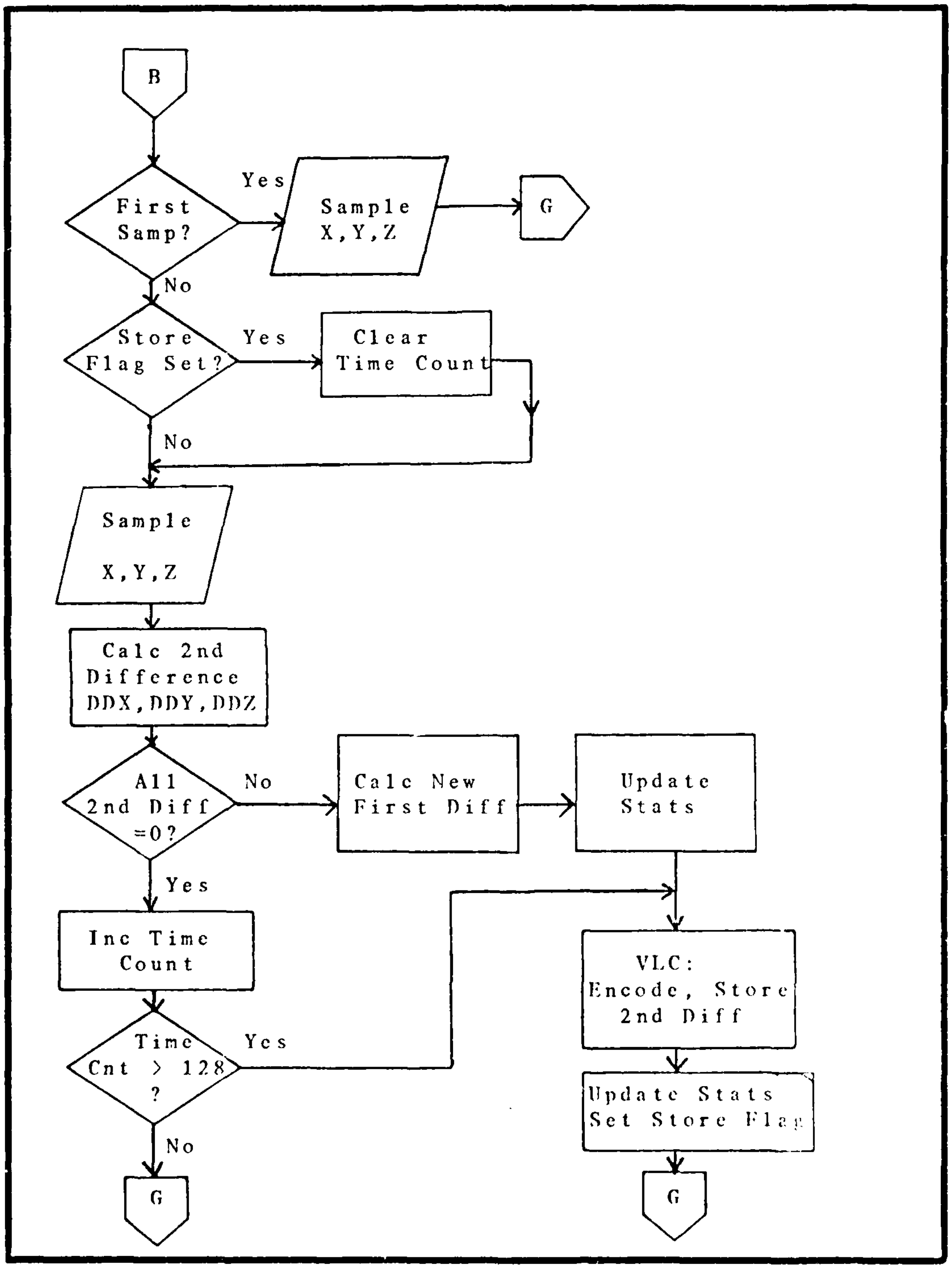


Fig 25b. TOLAN Functional Flowchart

Summary

This chapter has presented an overview of the complex software which makes up the EKG-DAAS. The reader is referred to the program listings in appendix C for a more thorough description of the program operation.

The software structure in the EKG-DAAS was written in a ''Top Down'' manner and all attempts have been made to document the operation of each routine. Since assembly language is difficult to read, this chapter was written to assist the reader in understanding the basic structure of the EKG-DAAS. The next chapter presents the results of the EKG experiment where data was collected, analyzed, and compressed by the EKG-DAAS.

V. Experimental Procedure, Data Analysis, and Results

This chapter presents the results of an EKG collection experiment where ''in vivo'' EKG data was taken from test subjects in real time. Data was taken and stored in both compressed and uncompressed formats for later analysis and reconstruction.

Chapter 5 is organized as follows. First the ''experimental'' setup is described along with a description of the EKG equipment, collection enviroment, and subject personnel. Next the parameters used to determine compression performance are defined followed by the analysis and results of the experimental data. Finally, the chapter concludes with a comparison between the results obtained using the Tolan compression algorithm and the estimated performance of the Dower technique. Discussion now turns to the experimental procedure.

Experimental Procedure

The data was taken from a set of nine fellow students during a laboratory course on electrocardiograms. The equipment used to produce the EKG was the model DR-12 research recorder built by Electronics for Medicine, Inc.(see appendix E). The DR-12 is a vintage medical recording system built in the late 1950's and is constructed with vacuum tube amplification circuitry.

The personnel used for test subject were all Air Force

officers in good physical health. No test subjects with obvious heart disease were used although siginificant EKG variations between subjects was noted (Fig 27-29). Output of the DR-12 was limited to one EKG signal which could be switched to any of the 6 ''limb leads'' (Ref 13:29-34) by controls on the DR-12. The electrodes of the EKG were applied to the wrists of the test subjects and in some cases, not all, an electrode jelly was applied to reduce skin-electrode resistance.

Since the EKG-Data Acquisition and Analysis System (EKG-DAAS) was configured for a 3 lead system, the X,Y,Z inputs were connected in common and the single signal available from the DR-11 applied to this connection. A Brush Instruments Mark II recorder and a Tektronics Model 465M Oscilloscope were used as analog output devices (Appendix E). The display instruments were connected in common with the A/D inputs as is shown in Figure 26.

Prior to the data recording session, the A/D was calibrated in accordance with the operating manual (Ref 8). The A/D was configured for a dynamic range of of -5.000 volts to +4.9976 volts with 2's complement binary representation.

The actual data collection proceeded as follows. First the subject was connected to the DR-12 and the amplitude of the resulting EKG signal adjusted to fall within the A/D dynamic range. A test run of all six limb leads (I,II,III,AVL,AVR,AVF) was then taken (without storage by

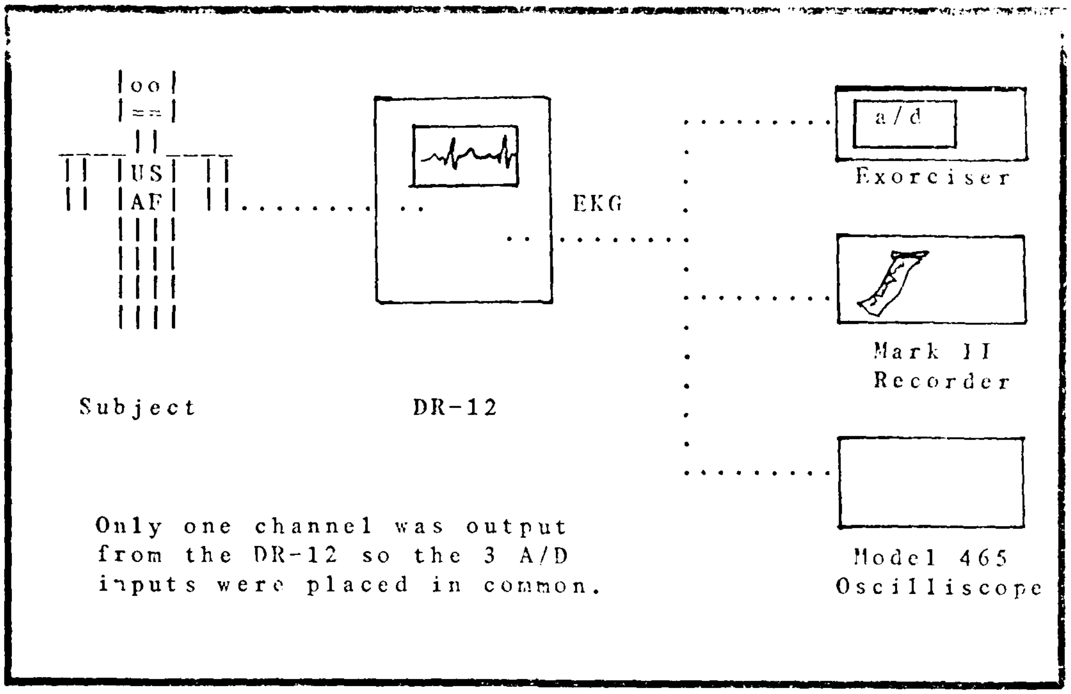


Fig 26. Experimental Data Collection Setup.

EKG-DAAS) and the lead with the least ''noisy'' signal was selected. EKG-DAAS was then executed and a data collection made <u>without</u> compression. This uncompressed data collection was followed as soon as possible with another ''run'' in which the Tolan compression algorithm was enabled. Following both data collections (uncompressed and compressed), the raw data traces from the Brush recorder were annotated with the time and subject and filed for later data comparison with the reconstructed waveform.

The data was stored on ''floppy'' diskettes and processed post collection for the entropy and maximum compression statistics. Before the results of these data

collections are presented, the compression measurement parameters will be defined.

EKG Compression Measurement Parameters

To permit determination of compression performance, a set of statistical parameters was calculated and saved during each data collection. These statistical parameters are described as follows:

Number of Samples. This statistic was saved to determine the total number of bits that were input to the compression stages. The total number of bits were calculated by (8 bits/sample)* 3 leads)*(num of samples/lead).

2nd Difference Frequency of Occurence. Four frequencies of occurrence tables were kept with double precision binary counters. Following the data collections, these $\Delta^2x, \Delta^2y, \Delta^2z, \Delta t$ histogram tables were input to a BASIC program (ENTROPY) where the entropy of second difference ''source'' was calculated.

Total Waiting Loop Counts During Collection. A counting loop was established in the TOLAN compression module which allowed determination of the percent of the sampling period used for the compression and statistics calculations. One circuit of this counting loop takes 46 machine cycles of the 6800 microprocessor. A count of the total number of loop cycles completed following the sampling/compression interrupts is kept in the collection statistics buffer.

Maximum Loop Count Per Interrupt. To offset the inaccuracy which would develop if the interrupt clock period changed between runs (or if the master clock in the Exerciser drifted), a loop count calibration was performed immediately before and after each collection run. This calibration was accomplished by performing 256 sequential interrupts with no interrupt processing except return-from-interrupt. The before and after calibration counts were then averaged and the maximum loop counts per interrupt calculated. Time Efficiency of the compression operation was

AD-A100 799 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 6/5
ANALYSIS AND PERFORMANCE EVALUATION OF ELECTROCARDIOGRAM DATA C--ETC(U)
DEC 80 M D TOWNSEND
UNCLASSIFIED AFIT/GE/EE/80D-46 NL

2 OF 3
AD A
100799

then calculated from the equation:

T.E.=(1-(Total Loop Count/((Num of Samples)*(Max Cnt))))*100

<u>Channel Maximums and Minimums</u>. The channel maximums and minimums were retained to allow determination if the analog inputs exceeded the A/D dynamic range.

<u>Number of Memory Bits Available</u>. This number was constant and was determined by the amount of Read/Write (RAM) memory available for data storage. For the current configuration of the Exorciser and the EKG-EXEC program this was 139248 bits (17406 bytes).

<u>Number of Bits Available to Variable Length Coder</u>. This counter measured the number of bits out of the data decorrelator and allowed calculation of the decorrelator's compression ratio (bits out/bits in).

<u>Number of Bits Used to Store Channel X,Y,Z</u>. These counters measured the number of bits used to store the data from the three input leads. This count is the number of <u>code bits</u> out of the variable length encoder.

<u>Number of Bits Used to Store Time</u>. This counter was identical to the channel counters above but measured the number of code bits used to store the Δt run counts.

<u>Total Compression Ratio Achieved</u>. This figure was calculated post collection by dividing the total code bits stored by the total R/W memory bits available.

The statistical data defined above was compiled by the EKG-EXEC program and is printed by the DISPLAY software module as illustrated in appendix D.

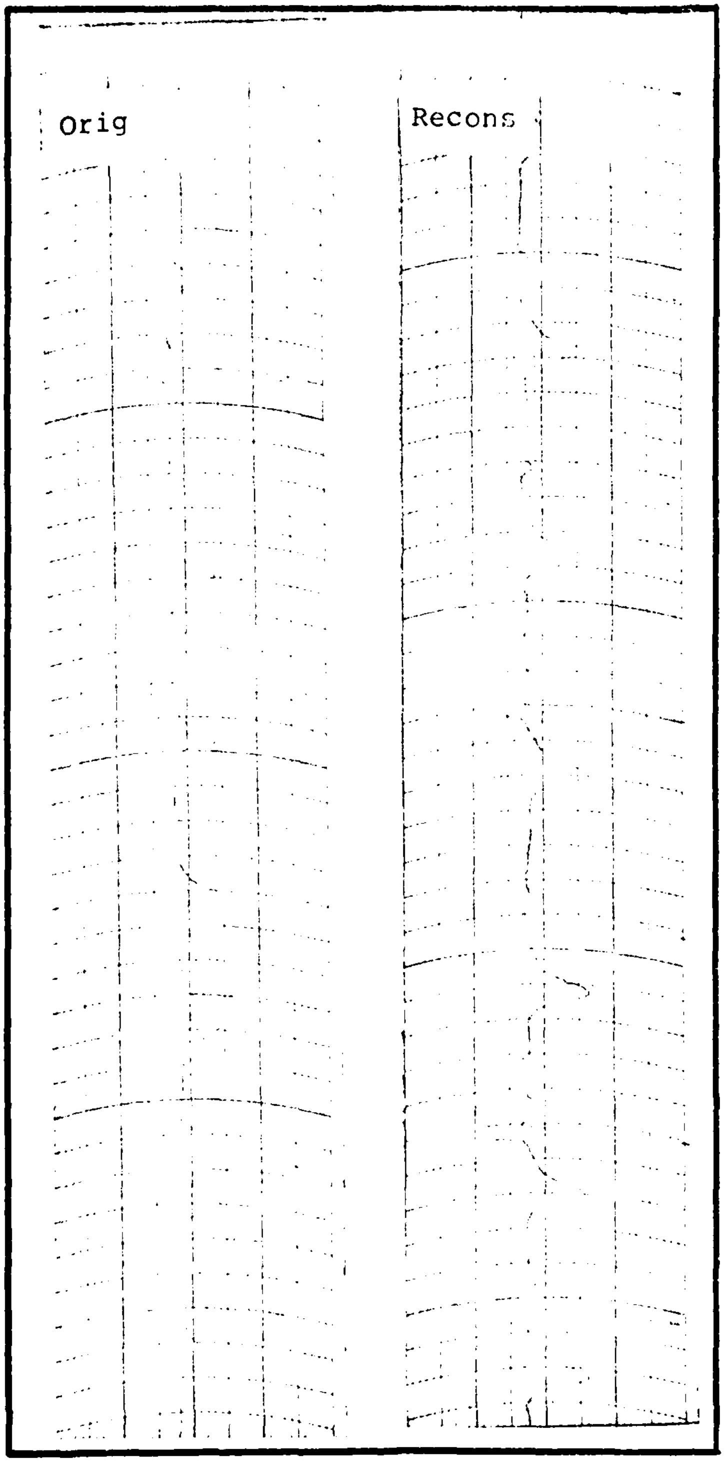


Fig 27. Original and Reconstructed EKG with Best Compression Ratio (2.260 : 1)

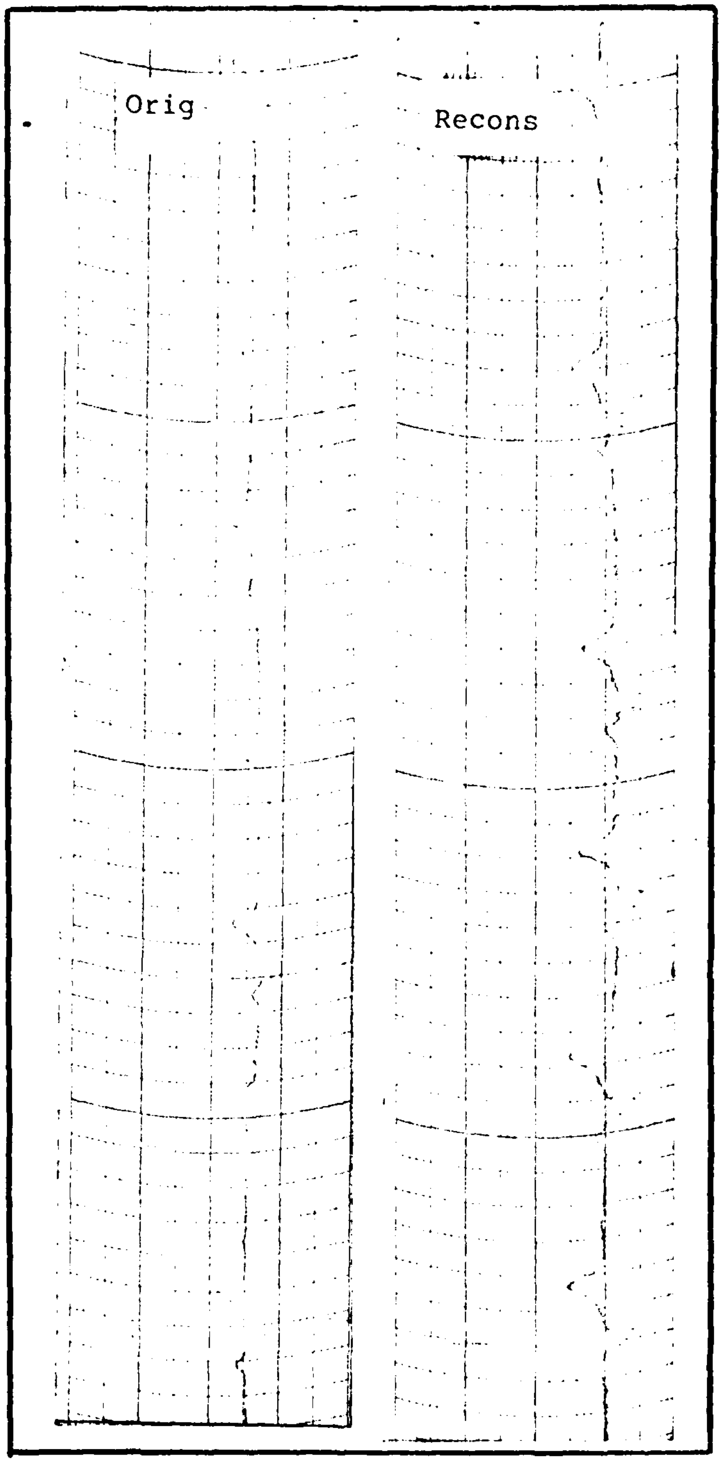


Fig 28. Original and Renconstructed EKG with Average Compression Ratio (1.60 : 1)

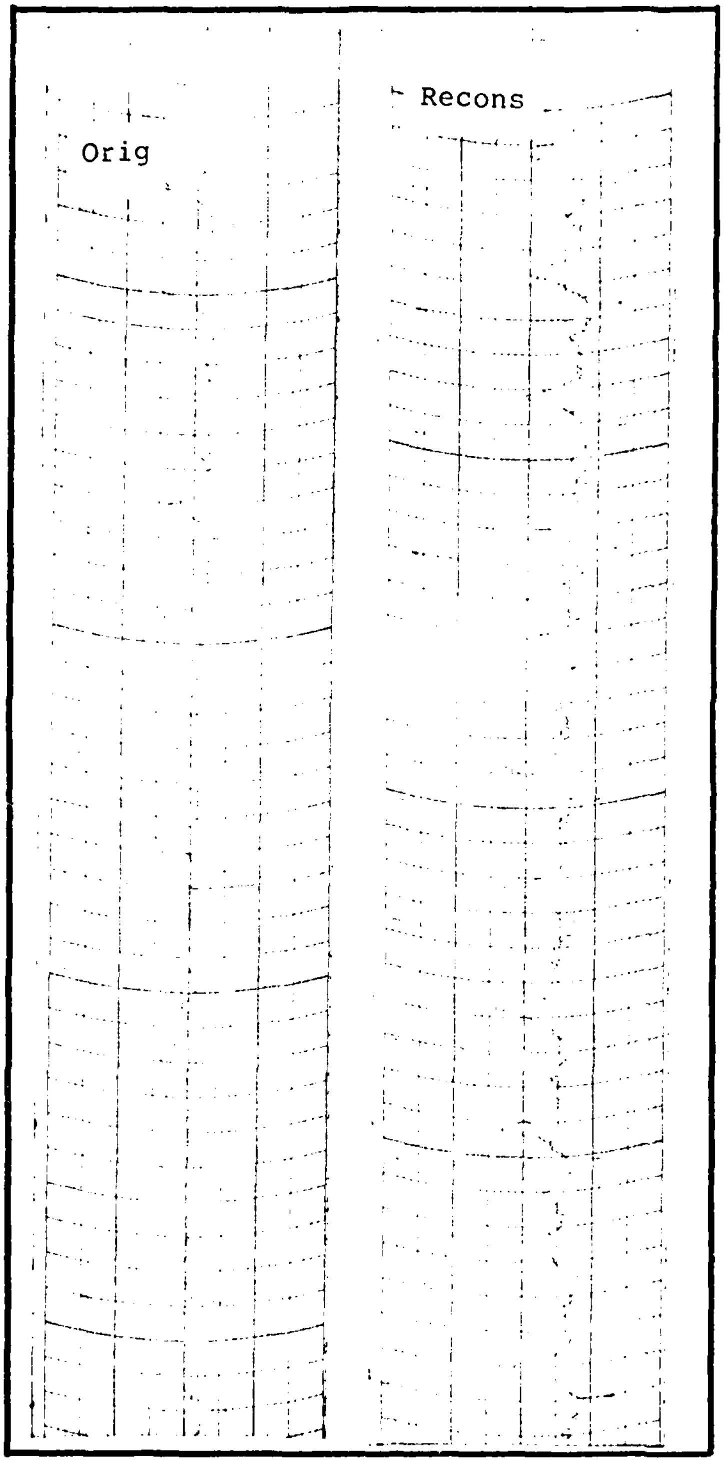


Fig 29 . Original and Reconstructed EKG with Worst Compression Ratio (1.252 : 1)

Data Analysis and Results

To determine the compression efficiency of a redundancy reduction EKG compression algorithm is, in general, a difficult job to accomplish. The reasons for this difficulty are: 1) the sampled EKG data sequence contains significant correlation (memory) between sample values making calculation of the absolute bound of the entropy extremely difficult (Ref 34:479-489) and ; 2) the techniques used for both the data decorrelator and entropy encoder vary significantly from algorithm to algorithm.

In this thesis, as is done in the literature (Ref 7,12,28), the assumption is made that the output of the decorrelator is ''almost decorrelated''. Decorrelated is not ''independent'' (unless the source was statistically gaussian), but it is assumed that true entropy of the 2nd difference sequence approaches the value which would be calculated by Eq.(1) (reproduced below).

$$\Delta^2 \text{ entropy} \cong -\sum -p_i \log_2 p_i \qquad (27)$$

(p_i is the probability of the i'th second difference).
This second difference entropy can then be used as an upper bound on the potential entropy encoding compression of the Tolan redundancy reduction technique.

The Δ^2 entropy values tabulated in Table II were calculated by Eq.(27). To calculate the approximate limit on the entropy encoder's compression ratio, the uncompressed data word length of 8 bits was divided by the ''lowest''

average code word (i.e., the 2nd difference entropy). A look at Table II shows that the entropy encoder compression ratio varied between 58% and 71% of this entropy bound. Since the Tolan variable length code is suboptimal, a lower efficiency is expected. Nevertheless, the Tolan entropy encoder performed more-or-less consistently across the data set. This last observation would imply that the data decorrelator influences the overall compression ratio more

Table II
Experimental Data Summary

Subject Id	2'nd Difference Entropy	Maximum Compression Possible (Approx)	Achieved 2nd Difference Compression	Percent of Max Encoder Comprs	Percent of Sample Interval	Achieved Total Comprs
TA1545T	3.2801	2.42 : 1	1.55 : 1	64.0%	93.9%	1.36 : 1
TA1548T	3.0171	2.63 : 1	1.61 : 1	61.2%	91.4%	1.50 : 1
TA1559B	3.016	2.64 : 1	1.72 : 1	65.1%	93.0%	1.53 : 1
TA1511S	3.323	2.39 : 1	1.65 : 1	69.0%	94.0%	1.43 : 1
TA1520B	2.601	3.06 : 1	1.81 : 1	59.1%	90.9%	1.72 : 1
TA1448L	2.930	2.72 : 1	1.73 : 1	63.6%	90.6%	1.60 : 1
TA1439S	3.267	2.43 : 1	1.63 : 1	67.1%	93.4%	1.43 : 1
TA1359P	2.487	3.22 : 1	1.88 : 1	58.3%	77.8%	2.26 : 1
TA1413L	3.783	2.10 : 1	1.50 : 1	71.4%	95.4%	1.25 : 1

than the entropy encoder.

As was described in chapter 3, the Tolan data decorrelator uses time compression in conjunction with a 2nd

difference operation. The Tolan time compression technique makes the assumption that the second difference value of zero occurs so frequently that the encoding of 0 would be less efficient than the storing of a zero value run counter. In the experimental situation in this thesis, signal noise was quite evident in the EKG traces (Fig 29). The sharp ''spikes'' induced by noise are accentuated by the second difference operation, hence a $\Delta^2 \neq 0$ was a common occurrence. This forced the storage of a lot of 7 bit time counters.

In 8 out of the 9 compression runs made, the frequent storage of time counts actually caused the Tolan data decorrelator to produce <u>negative</u> compression (i.e. more bits out than went in). Since the entropy encoder was producing a larger positive compression ratio, the overall compression figure remained positive. The effect of this operation is graphically illustrated in Figure 30.

A look at the original and reconstructed EKG traces (Fig 27-29) in conjunction with the data in Table II, shows that as the ''noise'' level increased on the signal the compression became progressively worse. Since it was concluded that the entropy encoder performed approximately the same across the data set, the degradation in total compression must be due, in large degree, to the degradation in the Tolan time compression data decorrelator performance.

A clear effect of the noise is illustrated in Figure

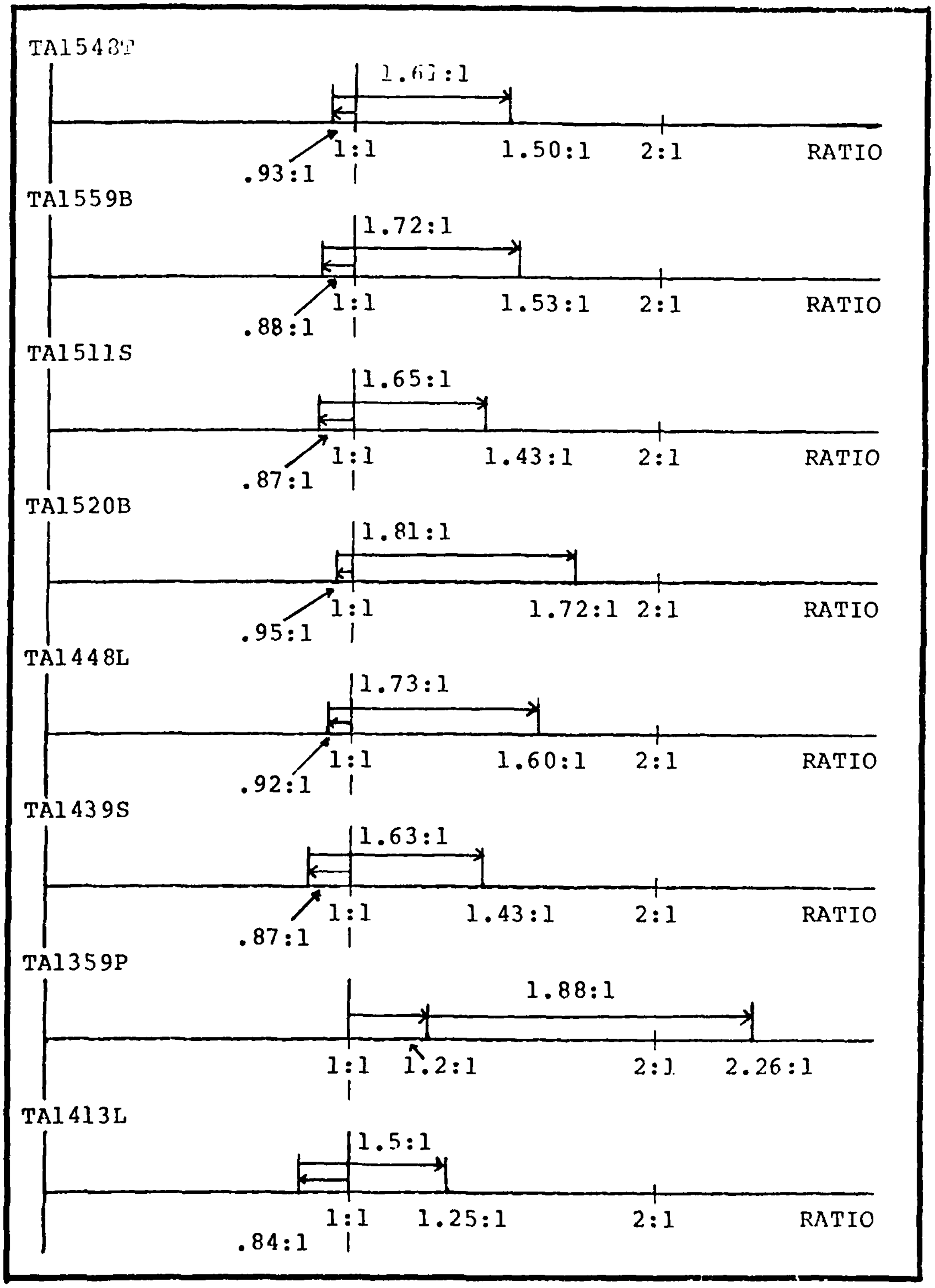
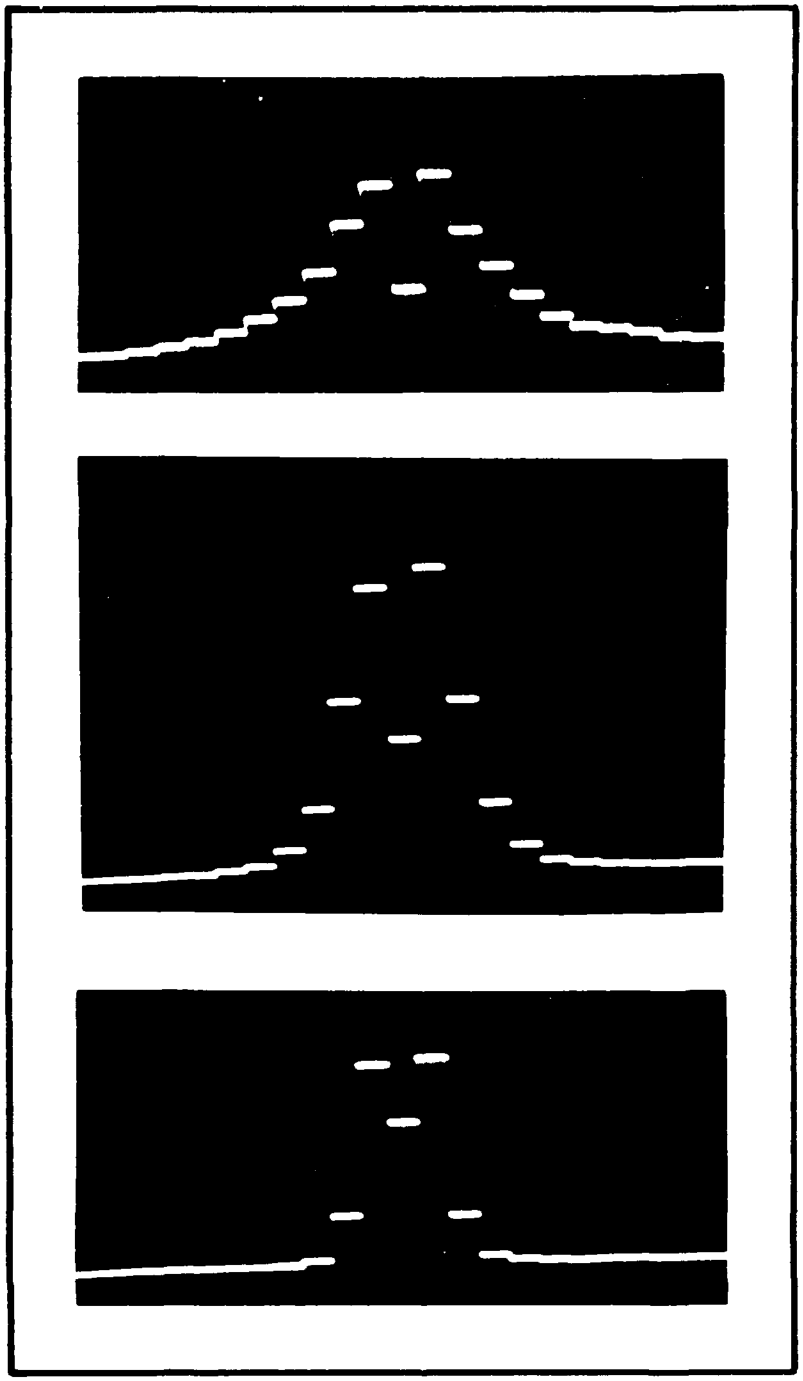


Fig 30 . Tolan Compression Ratio Breakdown.

31. From Figure 31, it can be seen that as the ''noise'' level increased the variance of the 2nd difference distribution became increasingly larger. As the distribution became less peaked, the efficiency of the variable length encoder decreased with a subsequent loss in achieved compression ratio.

The nonzero component at $\Delta^2 = 0$ in Figure 31 can be explained as follows. The decision was made early in the design of the EKG-DAAS that 8 bit data would be used versus 9 or 10 as recommended by the American Heart Association (Ref 3). This decision was made to simplfy the software (i.e. single precision could be used). Since the A/D converter has 12 bit resolution, the sample was rounded to 8 bits for uniform error distribution (Ref 24:424-432). Unfortunately because of this rounding action, small differences in the least significat bits of the A/D converter may have affected the rounding operation. Since the A/D cannot sample all three channel simultaneously, the probability that a ''noisy'' signal will change the least significant bit (or bits) is high. With a rounding operation, these changes may ripple to affect the least significant bit of the 8 bit data.

At most this effect would only cause a change in the least significant bit of the 8 bit rounded values. This would cause, however, the second differences between of channel X,Y,Z to be different even though they were connected in common. As was described in chapter 3, any of



the three leads with a non zero second difference forced the storage of all three data points. Hence many zeros data points were stored because of 2nd difference asymmetry which would not have occurred had the 12 bit A/D data been truncated instead of rounded.

This fact undoubtably affected the overall compression efficiency of the Tolan algorithm. Nonetheless, the distribution in Figure 31 would not have changed (except the zero value count) significantly and noise would still have broadened the 2nd difference distribution.

Tolan and Dower Performance Comparison

As has been reiterated several times in the text of this thesis, the author's original intention was to implement both the Dower and the Tolan algorithms for experimental test and comparison. Since time did not permit the Dower implementation, an experimental comparison was not possible.

As an attempt to compare the results of the Dower and Tolan compression routines, the results quoted from the papers by Dower,Berghofer,and Stewart (Ref 12,29) will be used. Dower states that his state space variable length encoder approaches the entropy bound of the second difference source ''with about 1.65% wastage'' (Ref 12:3). This value is significatly higher than the approximately 30% ''wastage'' observed with the Tolan variable length encoder.

Assuming similar operations of the second difference time compression (in this author's ''noisy'' enviroment), the Dower decorrelator is expected to have neqative compression. With the Tolan and Dower decorrelators assumed ''equal'', then the real gain of the Dower technique over the Tolan approach is in the VLC.

Assuming the Dower entropy encoder approached closely to the entropy bound, it can be extrapolated that the Dower algorithm would have achieved a maximum compression of approximately (1.2)*(3.22)=3.86:1 for the ''best'' EKG in Table II and Figure 30. The worst compression ratio would have been (.84)*(2.1)=1.76*1 for the worst (noisest) EKG.

Chapter 5 Summary

This chapter began with a description of the EKG collection experiment. Although the EKG apparatus was limited to one channel, sucessful collection and compression of EKG data was performed. Analysis of the data revealed that time compression is inefficient in a ''noisy'' enviroment and that ''rounding'' of the 12 bit A/D samples in conjunction with placing all three sample lead in common accentuated the degradation caused by the time compression data decorrelator. Nonetheless, the Tolan algorithm did achieve an overall positive data compression figure but significantly lower than the 9:1 ratio achieved by Ruttiman and Pipberger (Ref 28) or the average value of 7.3:1 reported by Stewart,Berghofer, and Dower (Ref 29). Finally

it was heuristically shown that if the Dower compression algorithm lived up to the statements by Dower, then a compression ratio gain of 3.86:1 to 2.26:1 could have been achieved with the Dower EKG data compression technique. This thesis will now proceed to provide conclusions and recommendations

VI. Summary, Conclusions, and Recommendations

Summary and Conclusions

This thesis has investigated the field of electrocardiogram data compression with the objective of evaluating compression algorithms on a 6800 microprocessor based computer system. Accomplishment of this goal required the construction of the EKG-Data Acquisition and Analysis System utilizing the Motorola Exorciser microcomputer.

To determine those EKG compression algorithms which had potential for Exorciser implementation, a literature search was made to locate EKG data compression techniques. In addition to the literature search, personal correspondence (Ref 11,31) yielded several EKG compression algorithms. The results of this research is presented in chapter 2.

Since thesis requirements dictated the need for an online, real time data compression algorithm, only the fastest EKG compressors could be considered. Two routines were selected for detailed analysis and inplementation. These two compression algorithms (Tolan and Dower) were discussed at length in chapter 3.

It was deduced from the methodology of the two data compression techniques that the Dower algorithm would perform better than the Tolan procedure. To test this

hypothesis, the EKG-DAAS was constructed and the Tolan algorithm implemented. The implementation of the EKG-DAAS is documented in chapter 4.

Time constraints prohibited completion of the Dower compression algorithm, but data was sucessfully compresssed, analyzed, and decompressed with the Tolan algorithm. The results of this analysis are presented in chapter 5.

The conclusions of this research effort are as follows. First, EKG data compression can be accomplished in real time by a microprocessor based computer system. The Exorciser is a slow microcomputer (1 MHz cycle time) yet it was still possible to implement the Tolan algorithm with a 500 Hz sample rate. The second conclusion is that signal noise can dramatically affect the efficiency of the EKG routines in the same class as the Tolan algorithm. The expense and implementation difficulties of prefiltering, low electromagnetic noise environment, and proper EKG lead attachment are well worth the gain in compression achieved. Finally, a software project of this magnitude should not be attempted totally in assembly language. Although assembly language offers the greatest flexibility and speed, algorithm implementation and debugging efforts are enormous. A high order language would have allowed this author to complete his original thesis objectives.

Recommendations

Several recommendations are offered for further study in microcomputer based EKG data compression. First, implementation of the EKG compression algorithms using a 16 bit microprocessor (e.g. 6 MHz Intel 8086) would permit an order of magnitude improvement in speed of execution. Hardware multiply and divide along with 16 bit arithmetic registers would permit easy implementation of the Dower algorithm and would even make use of the Transform compressors (i.e. FFT) feasible. Second, futher study is needed on determination of decorrelator inefficiency on the overall data compression. A large study of different decorrelators such as 1, 2, 3 difference operations with and without time compression is needed. Next, an EKG compression algorithm implemented using the Ruttiman and Pipberger 2nd order interpolator (Ref 28) for the decorrelator along with the Dower variable length encoder (Ref 12) should be built. This combination should prove to be very efficient. Finally, programming and experimental testing should be done on a full scale microcomputer development system, complete with a high order language, A/D-D/A capability, and flexible disk file manipulation software. Such a system is the Zilog MCZ1/25 microcomputer resident here at A.F.I.T.

Bibliography

1. Ahmed, Nasir et al. ''Electrocardiographic Data Compression Via Orthogonal Transforms,'' IEEE Transactions on Biomedical Engineering, BME-22, (6) : 484-487 (November 1975).

2. Ahmed,N and Rao,K.R. Orthogonal Transforms for Digital Signal Processing. New York: Springer-Verlag, 1975.

3. American Heart Association, Rep. Comm. ECG. ''Recommendations for Standardization of Leads and Specifications for Instruments in Electrocardiography and Vectorcardiography,'' Circulation, 54: 11-31 (1975).

4. Berger, Toby. Rate Distortion Theory: A Mathematical Basis For Data Compression. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1971.

5. Blasbalg, H. and Van Blerkom, R. ''Message Compression,'' IRE Transaction on Space Electronics and Telemetry, 8 : 228-238 (September 1962).

6. Box, E and Jenkins, G.W. Time Series Analysis: Forecasting and Control. San Francisco, California : Holden-Day Inc, 1976.

7. Cox, J.R. and Ripley, K.L. ''Compact Digital Coding of Electrocardiographic Data,'' Proceedings Sixth Hawaii International Conference on System Sciences. 333-336. Honolulu, Hawaii : University of Hawaii, January 9-11, 1973.

8. DATEL SYSTEMS,Inc. Sinetrac Series Model ST-6800 A/D-D/A Peripheral Systems Instruction Manual, Part No. 58-12140-25, Mansfield, Mass. January, 1979.

9. Davenport, W.B. Probability and Random Processes: An Introduction For Applied Scientists and Engineers. New York : McGraw-Hill Book Company, 1970.

10. Davisson, Lee D. and Gray, Robert M., editors. Benchmark Papers In Electrical Engineering and Computer Science, Volume 14 : Data Compression. Stroudsburg, Pennsylvania : Dowden, Hutchinson,and Rose, Inc., 1976.

11. Diver, Tom. Project Engineer. Personal Correspondence. Marquette Electronics, Inc.,Milwaukee, Wisconsin, 15 May 1980.

12. Dower,RG. with appendix by Berghofer,D. ''Optimal ECG Data Compression,'' working paper submitted for publication in Proceedings of the Engineering Foundation Conferences: Computerized Interpretation of the ECG V. Asilomar, California. April 27-March 2, 1980.

13. Dubin,Dale, M.D. Rapid Interpretation of EKG's (Second Edition). Tampa, Florida: COVER Publishing Company, 1970.

14. Frank, Ernest. ''An Accurate, Clinical Practical System For Spatial Vectorcardiography,'' Circulation, 13 : 737-749 (May 1956).

15. Gray, R.M. and Davisson, L.D. ''A Mathematical Theory of Data Compression?'', Benchmark Papers In Electrical Engineering and Computer Science, Volume 14 : Data Compression, edited by L.D. Davisson and R.M. Gray. 21-25. Stroudsburg, Pennsylvania : Dowden, Hutchinson, and Rose, Inc., 1976.

16. Huffman, David A. ''A Method For The Construction of Minimum Redundancy Codes,'' Proceedings of the IRE, 40 : 1098-1101 (September 1952).

17. Jelinek,Frederick. ''Buffer Overflow in Variable Length Coding of Fixed Rate Sources,'' IEEE Transactions on Information Theory, IT-14 (3) : 490-501 (May 1968).

18. Makhoul, John. ''Linear Prediction: A Tutorial Review,'' Proceedings of the IEEE, 63 (4) : 561-580 (April 1975).

19. McEliece, Robert J. The Theory of Information and Coding : Encyclopedia of Mathematics and its Applications, Volume 3. Gian-Carlo Rota, editor. Reading, Massachusetts : Addison-Wesley Publishing Company, 1977.

20. McFee,R. and Baule,G.M. ''Reasearch in Electrocardiography and Magnetocardiography,'' Proceedings of the IEEE , 60 (3) :290-303 (March 1972).

21. McGibbon, C.I. ''Data Communication Networks of Interest to Medical Information Processing,'' Proceedings of the IFIP TC 4 Working Conference on Optimization of Computer ECG Processing, edited by H.K Wolf and P.W. MacFarlane. 253-254. Halifax, Nova Scotia, Canada. June 5-7, 1979.

22. Motorola Inc., M6800 EXORciser Users Guide, Phoenix,Arizona: Second Edition, 1975.

23. Midwest Scientific Instruments, Operating and Assembly Manual, The FD-8 Floppy Disk Memory System, Olathe, Kansas.

24. Oppenheim, Alan V. and Shafer, Ronald W. Digital Signal Processing. Englewood Cliffs, N.J. : Prentice-Hall,Inc. 1975.

25. Papoulis, Athanasios. Probability, Random Variables, and Stochastic Processes. New York: McGraw Hill Book Company, 1965.

26. Rautaharju,P.M., Warren, J., and Wolf, H. ''Waveform Vector Analysis of Orthogonal Electrocardiograms: Quantification and Data Reduction, Journal of Electrocardiography, 6: 103-111 (March 1973).

27. Ristenbatt, Marlin P. ''Alternatives in Digital Communications,'' Proceedings of the IEEE, 61 (6):703-721 (June 1973).

28. Ruttimann, Urs E. and Pipberger, Hubert V. ''Compression of the ECG by Prediction or Interpolation and Entropy Encoding,'' IEEE Transactions on Biomedical Engineering, BME-26, 11 : 613-622 (November 1979).

29. Stewart,D.,Berghofer,D. with appendix by Dower,RG. ''Data Compression of ECG Signals,'' Proceedings of the Engineering Foundation Conferences, Computerized Interpretation of ECG IV. Asilomar, California. January, 1979.

30. Shannon, Claude E. ''A Mathematical Theory of Communication,'' Bell Systems Technical Journal, 27 :379-423 (July 1948).

31. Tolan, Gil. (MD),Lt.Col.USAFMC,Personal Correspondence, USAF School of Aeorspace Medicine, Brooks AFB, Texas. April,1980.

32. Tompkins, W. and Webster, J.G., editors. Design of Microcomputer-Based Medical Instrumentation. Englewood Cliffs, N.J.: Prentice-Hall, Inc.,1980.

33. TRW Systems Group, Houston, Texas. ''Investigation of Data Compression Techniques,'' Final Report, NASA-CR-115177. September, 1971, N71-35323.

34. Viterbi Andrew and Omura, Jim. Principles of Digital Communication and Coding. New York : McGraw Hill Book Company, 1979.

35. Womble, Edward M. *et al.* ''Data Compression For Storing and Transmitting ECG/VCG's,'' *Proceedings of the IEEE*, *65* (5) : 702-706 (May 1977).

36. Ziemer, R.E. and Tranter, W.H. *Principles of Communcations: Systems, Modulation, and Noise*. Boston, Massachusetts : Houghton Mifflin Company, 1976.

Appendix A

The Electrocardiogram

Introduction

The electrocardiogram (or EKG) is a record of the electrical activity of the heart as measured from the body surface. The magnitude, shape,and timing of the electrical potentials generated by the heart reveal a great deal of information concerning the health of the cardiac system. This appendix will describe how the electrical signals from the heart are generated, how the cardiac cycle is coordinated and controlled, and finally how the EKG can be used as a diagnostic tool.

The Physiology and Electricial Characteristics of the Heart

The heart (Fig. A1) is an organ about the size of a fist with four main pumping chambers and a specialized electrical conduction system. At the top are two thin walled pumps called the atrium which prime the main pumps of the heart, the ventricles. The ventricles are separated by a thick wall of muscle tissue called the septum. Blood from the right ventricle goes to the lungs and blood from the left ventricle goes to the rest of the body.

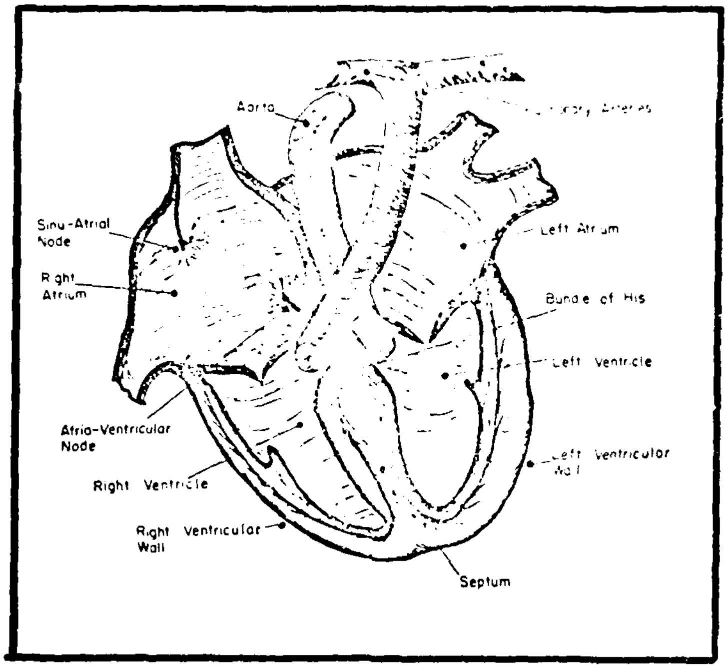


Fig A1. The Heart Cross-Section (From Ref 20:291).

During the resting state between contractions, the cells of the heart are ''polarized'' with an electrical potential existing between the inside (-) and the outside (+) of the cell. This potential is generated by an ionic gradient across the cellular membrane and is normally maintained for approximately .2 to .4 of a second before spontaneous ''depolarization'' occurs. Depolarization (caused by an inrush of sodiun ions into the cell) induces the cell to contract for approximately 1/4 of a second. Because all heart cells contain specialized conducting fibers, the depolarization of one cell initiates the depolarization of neighboring cells and a ''wave of excitation'' sweeps across the myocardium (heart) at a rate

of about 1 meter per second.

The voltages measured at the body surface are the superposition of thousands of heart cells depolarizing (or polarizing) as the wave of excitement flows through the myocardium. Early work by Wilson (Ref 20:292) showed that the heart could be represented by an equivalent electrical dipole whose vector orientation sweeps through a closed loop during one cardiac cycle. In simple terms, as the wave of excitement flows toward a positive skin electrode, a positive slope is generated on the EKG record representing the projection of the heart vector onto the axis of the EKG lead.

To observe this sweeping dipole vector, Wilson developed the 12 lead EKG system in almost univerisal use today. This system (Fig. A2) attempts to measure the heart vector from a variety of vantage points in hope of determing the actual direction of propagation of the wave of excitement. Unfortunately, the leads of the Wilson system are not orthogoal and reconstruction of the actual heart vector orientation and amplitude is difficult. To overcome this problem, Frank (Ref 14:737-749) developed the vector cardiogram which combines 7 leads in a summing network to produce three orthogonal components of the heart vector. Frank VCG systems are popular in heart diagnosis and research because all of the information is contained in three leads of data versus 12.

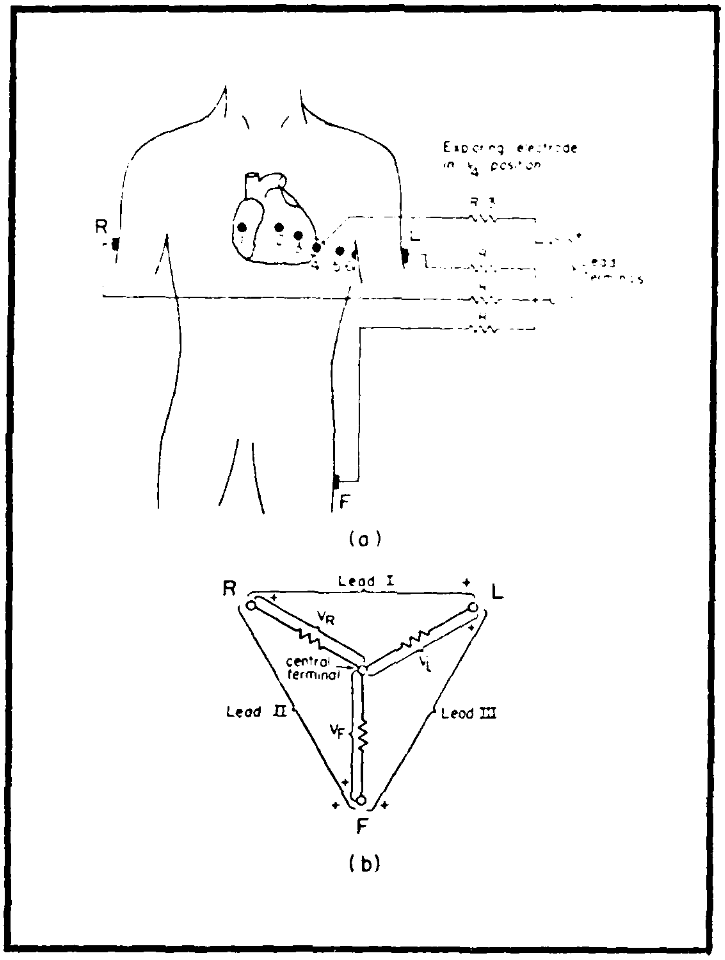


Fig A2. Wilson EKG Electrode System (From Ref 20)

The Cardiac Cycle

For effective pumping action, the heart muscle must contract in a controlled, coordinated way. This means that the stimulating wave of excitation must follow a well defined conduction path to allow the heart to contract in the most efficient manner. In ''normal'' hearts, the contraction sequence begins in the right atrium where

special cells in an area known as the Sino-Atrium (SA) node spontaneously depolarize faster than the rest of the heart tissue. The SA node hence initiates a wave of excitation which covers the atrium in about 80 milliseconds. This atrial contraction produces the electrical signal called the P wave (Fig. A3) on a typical EKG record.

The wave of excitation started by the SA node then reaches another specialized receptor known as the Atrial-Ventricular (AV) node. Here connecting fibers delay the excitation impulse for about 50 milliseconds to allow the ventricles to fill with blood. After the 50 millisecond delay, the excitation signal is relayed to special conducting fibers in the septum known as the Bundle of His. These conducting fibers rapidily (30 ms) transmit the excitation wave to the interior (endocardium) wall of the ventricles where the wavefront propagates radially to the outer wall (epicardium) in another 30 milliseconds. The ventricular wave of excitation produces the QRS waveform complex seen on the EKG.

Following the ventricular contraction, the muscle cells of the ventricles repolarize over a period of 100 milliseconds. No muscular action is occuring but the EKG responds to this electrical activity and the T wave is noted on the EKG record. Atrial repolarization occurs during the QRS hence it is generally invisible on the EKG.

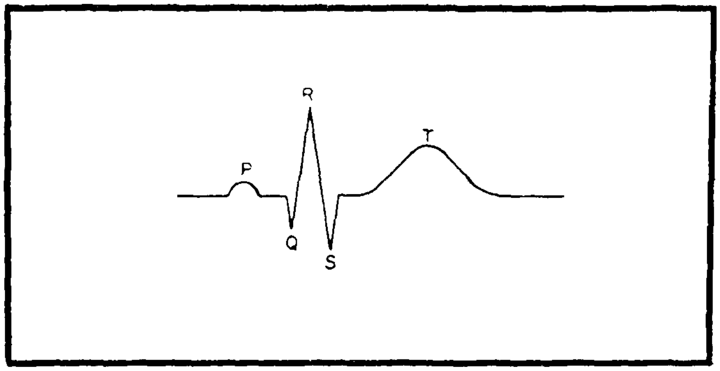


Fig A3. Typical EKG Waveform (From Ref 20)

Finally the heart rests for approximately .2 of a second and the cycle starts again. The above cardiac cycle is typical of a healthy heart. Disease, however, can effect this sequence dramatically.

<u>Heart Disease and the EKG</u>

The variety of ailments which plauge the human heart are numerous and no attempt will made to describe the spectrum of diseases possible. There are, however, several common heart defects which routine EKG analysis usually detects. These include premature contractions (Atrial and Ventricle), bundle branch blocks, hypertrophy, and infarction.

<u>Premature Contractions</u>. Premature contractions are caused by the spontaneous depolarization of heart tissue outside of the SA node. This depolarization initiates a wave of excitement causing the atrium,or more notably, the

ventricles to contract out of rhythm with their normal cycle. Points where this ''unscheduled'' depolarization occurs are called ectopic foci and can arise because of infarct damage (to be disscussed later), coronary heart disease producing oxygen starvation and a number of other causes. Premature Ventricular Contractions (PVC's) are highly visible on the EKG record. Normally the ventricles contract simultaneously and the voltage vectors generated tend to cancel keeping the QRS amplitude relatively small. A PVC, however, causes depolarization of one ventricle before the other generating an unbalanced, hence larger, voltage output.

<u>Bundle Branch Blocks</u>. A bundle branch block is caused by a block of the impulse of the right or left Bundle Branch. This causes a delay in the transmission of the stimulation impulse to ventricle blocked and forces the two ventricles to contract at slightly different times. This difference in ventrical contraction time shows up on the EKG as a double humped QRS.

<u>Hypertrophy</u>. Hypertrophy is an enlargment of one section of the heart muscle tissue. This enlargment affects the duration and strength of the wave of excitement and is visible on the EKG record as a diphasic trace if atrial hypertrophy is present. If ventricular hypertrophy exists, the QRS amplitudes are much larger than normal due to the fact that more tissue is depolarizing.

Infarction. Myocardial infarction is an injury to the heart t... caused by an occlusion of a coronary artery. An area of the heart is then without a blood supply and often permanent damage occurs. This heart disease is often the one most commonly called a ''heart attack'' and, as is well known, is many times fatal. If a person survives the orginal ''attack'', then this permanent damage shows up in the EKG as a change in the QRS and T waves. This change occurs because the infarcted tissue no longer responds to the excitation wave and the wave moves around, not through, the damaged tissue.

Summary.

This appendix has briefly examined the physiology of the heart and discussed how the myocardial tissue generates the electrical fields measured by the electrocardiogram. Though EKG analysis has been practiced for fifty years, intense research continues in improving EKG diagnosis. Dramatic improvement in computer aided EKG analysis and better understanding of the electro-physiology of the heart is leading to improved cardiac health care worldwide.

The heart diseases discussed above are only a small subset of the problems which can afflict the human heart. Should the reader desire a more thourgh background on heart physiology and cardiac disease, Dubin's book (Ref 13) is highly recommended. This programmed text carefully leads the reader through EKG analysis and is easily read. For a

more firm background on the electro-physics of the heart, the article by McFee and Baule (Ref. 20) provides a good tutorial review on EKG history and research.

Appendix B

Fundamentals of Information Theory

Introduction

In 1948, Claude Shannon published a classic paper (Ref 30:379-423) titled ''A Mathematical Theory of Communication'' in which he laid the foundation of modern information theory. Shannon used his ''information theory'' to describe, mathematically, the interrelationships between the components of a ''typical'' communication system as illustrated in Figure B-1.

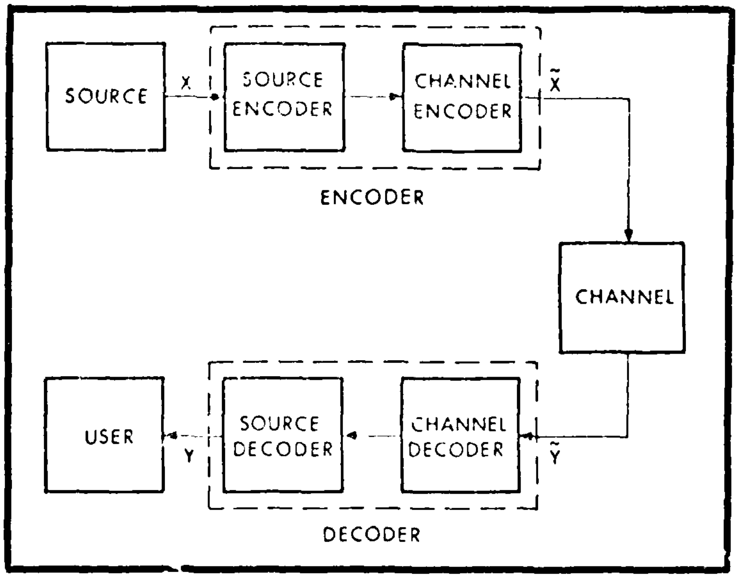


Fig B-1. The communication system model (From Ref 4).

This appendix is written as a basic tutorial on information theory, and is intended to acquaint the reader

with the terminology used in describing the data compression techniques in chapters 2 and 3. The appendix begins with a review of the information source, proceeds to a discussion of the transmission channel and Shannon's rate distortion theory, and concludes with a description of the system encoder/decoder.

Information Source

For his initial analysis, Shannon proposed modelling the information source as a discrete stochastic process whose output is governed by known statistics. As pointed out by Davisson and Gray (Ref 10: 2-4), the discrete-time model was commonly used for any or all of the following reasons: (1) digital communication links have become common place; (2) a continuous time process can be modelled as discrete by sampling, orthogonal function expansion, or waveform segmentation; (3) greater simplicity.

The source is characterized by a finite set of possible outcomes known as its alphabet A. The occurrence of a particular alphabet symbol is governed by probabilistic descriptors (i.e. probability density functions) and it is assumed that the source produces only one symbol from the alphabet every T_s seconds. Hence the information source has a symbol rate of $R_s=1/T_s$ symbols per second.

The next question of interest is how much information is conveyed by the occurrence of a given source symbol? If X is a discrete random variable occurring at time t, and x is

an element of A, then the random variable <u>self information</u> can be defined. That is

$$I(x) = -\log p\{X=x\} \qquad (B.1)$$

where $p\{X=x\}$ is the probability that $X=x$.
According to this description, the less probable an event is, the more information is conveyed when it occurs. The base of the logarithm is unspecified, but in this thesis it is assumed to be base 2. Hence the occurrence of symbol x reveals I(x) <u>bits</u> of information.

The amount of information received per observation is of interest, but one would like a measure of the ''uncertainty'' or ''randomness'' of the source. If the stochastic process defining the source is considered stationary (a pretentious assumption but one generally made) then the output of the source is a sequence of random variables with identical probabilistic descriptors. The probabilistic descriptor will be defined as {u} and could represent the moments of the random variable or its probability density function (PDF). If, in addition, the source is considered ergodic, then statistical averages equal time averages and calculation of the set {u} is greatly simplified.

Given that the discrete process is ergodic, or at least stationary, then the source output at any time is described by the random variable X with range $A=\{x(1), x(2), \ldots x(n)\}$. The measure of the ''uncertainty'' or ''randomness'' is

defined as the <u>entropy</u> of the source and is given by the [illegible]tion

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i \qquad (B.2)$$

where p_i is the probability of occurence of the discrete value x_i. If $p_i = 0$, then the term $\log_2 1/p_i$ is defined equal to 0 (i.e. no contribution to the entropy). Should the range A be of infinite extent (i.e. a continuous source), then the above series may not converge nor would p(x) necessarily be defined. In this case, H(X) is defined as positive infinity.

As example (Ref 19:15), let X represent the outcome of a single roll of a fair die. Then A={1,2,3,4,5,6} and p_i =1/6 for each i. Here $H(X) = \sum_{6} 1/6 \log_2 6 = 2.58$ bits.

In the above example, the statistics governing the occurrence of a given outcome were uniform. This represents the ''most random'' case with a resultant maximum of the entropy function. Should the die be ''loaded'', then the predicted outcome is ''less random'' and hence the value of H(X) would be reduced.

The next important component in Shannon's communication model is the transmission channel.

Transmission Channel

The transmission channel is also assumed to be a discrete time, finite alphabet device which accepts and transmitts to the receiver one symbol in a finite alphabet B each T_c seconds. The alphabet B is often binary, hence the dimension of the symbol space (defined as $||B||$) is 2 and B={0,1}. The transmission rate of the channel is defined as $R_c=1/T_c$ channel symbols (bits if binary) per second. If the output of the source process is defined as {X}, and the channel is ''noisy'', then the received symblol {Y} might not equal the transmitted symbol {X}. A noiseless channel transmitts symbols with no error in which case {X}={Y}.

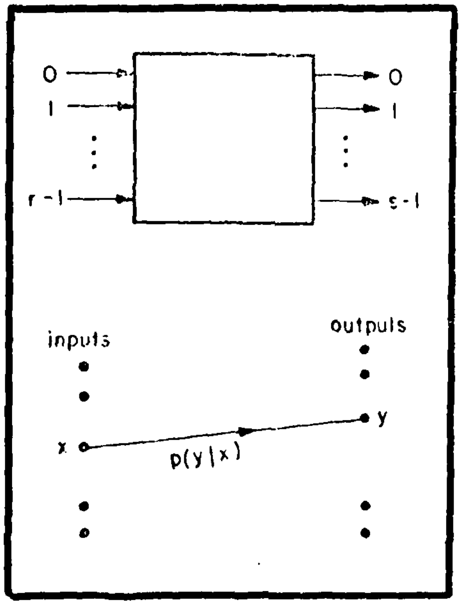


Fig B-2. A discrete memoryless channel (From Ref 19:19)

For simplicity, let the channel be modelled as a

discrete memoryless channel (DMC) as depicted in Figure B-2. The DMC is described mathematically by a conditional probability which relates the chance that a given output y was the result of a given input x. If the input to the channel is a random variable X, and the output is a random variable Y, then a quantity called conditional entropy can be defined. The equation defining conditional entropy is:

$$H(X|Y) = \sum_{x=1}^{n} \sum_{y=1}^{n} p(x,y) \log_2 \frac{1}{p(x|y)} \qquad \text{(B.3)}$$

For a given pair X,Y of random variables, $H(X|Y)$ represents the amount of uncertainty remaning about X after Y has been observed.

Now since $H(X)$ represents the uncertainty about X before X is known and $H(X|Y)$ represents the uncertainty after, the difference $H(X)-H(X|Y)$ must represent the amount of information provided about X by Y. This quantity is called the mutual information between X and Y, and is denoted by :

$$I(X;Y)=H(X)-H(X|Y) \qquad \text{(B.4)}$$

With the above definitions in hand, the most important quantity of a communications channel can be described; that quantity is the channel capacity. Channel capacity is

defined as the maximum amount of information, per unit of time, which can be ''reliably'' transmitted over the channel. That is :

$$C=\max\{I(X;Y)\} \tag{B.6}$$

Channel capacity is closely related to another important parameter of a communication system known as the rate-distortion function.

Rate-Distortion. Shannon postulated the existence of a mathematical distortion measure, d(X,Y), to measure the distortion or loss resulting if a source symbol X is reproduced as Y. Unfortunately this abstract distortion measure is difficult to quantify. As stated by Berger (Ref 4:6), ''the unavailablity of a distortion measure that is both physically meaningful and analytically tractable constitutes one of the major obstacles to progress in (communication) system design.''

Assuming one has such a distortion measure, then associated with most source-user pairs is a function R(D) called the rate distortion function. The rate distortion is important in that it gives the designer a mathematical tool to measure the amount of distortion that can be expected for a given transmission rate. A communication system can achieve a given fidelity D if and only if the capacity C exceeds R(D). Hence R(D) is the effective rate at which the

source produces information subject to the constraint that the user can tolerate an average distortion of D.

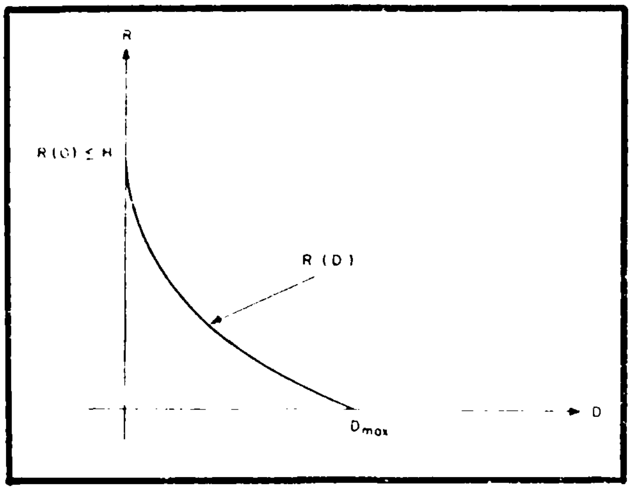


Fig B-3. A typical rate distortion function (From Ref 4:7).

The simplest source-user pair is a discrete memoryless source (DMS) and a single letter fidelity criteria. The DMS produces statistically independent, identically distributed, discrete random variables. Assuming a single letter fidelity criterion, every time the system presents the letter y to the user when the source output was actually x, a nonnegative penalty p(x,y) is determined. The rate-distortion function for the above case is plotted in Figure B-3. As can be seen from the Figure, R(0) is equal to H(X). This last result leads to Shannon's famous channel coding theorem which states that if the symbol rate is less than the channel capacity, it is possible to transmit with perfect fidelity. That is, if the entropy of the source,

H(X), is less than or equal to the channel capacity C, then transmission with zero error can be obtained. This amazing result is not without cost, however. To reduce the source entropy (as seen by the channel) requires <u>coding</u>.

<u>Encoder</u>

In most circumstances, the output of the information source is not suitable for direct input into the channel. To match the source to the channel, much like matching impedances in circuit theory, is the job of the encoder.

The encoder incoporates all of the functions which process the source data for transmission over the communication channel. This includes coding, analog-to-digital conversion, and modulation. In order to transmit without loss, integers K and L must exist such that $KTs = LTc$. This guarantees that the received sequence {Y} has the same <u>symbol rate</u> as the transmitted sequence {X}.

As illustrated in Figure B-1, the endoder is divided into two functional subunits. The first of these subunits is the source encoder.

<u>Source Encoder</u>. Source encoding is the operation by which the source output is mapped into an alterate symbol set with the goal of reducing source sequence dependence (i.e. reduce redundancy). Source coding is the transformation in which <u>data compression</u> occurs.

The first source codes, as conceived by Shannon, were <u>block codes</u> where blocks of source symbols were mapped into a single representative channel symbol. This type of

encoding is simple, but efficient only if the source repeats a given, constant length, sequence of symbols on a frequent basis. Another type of source coding is variable length coding.

In _variable length codes_, those source symbols which occur most frequently are assigned the shortest channel codewords.

One of the most popular variable length codes in use was originally proposed by Huffman (Ref 16:31-34) in the early fifties. This code is in a class known as _uniquely decodable_ (UD) codes. UD codes imply that the codewords, regardless of length, are unique sequences. Hence decoding is instantaneous upon codeword reception(i.e. the decoding does not depend upon reception of the next codeword). McEliece has shown (Ref 19:244-245) that the Huffman code is _optimal_ in the class of UD source codes.

Many techniques for source coding are available. Chapter 2 of this thesis discusses some approaches used in ''compressing'' electrocardiogram waveforms.

The otuput of the source coder is generally a sequence of discrete symbols (i.e. binary 1|0) suitable for futher processing by the channel coder.

Channel Encoder. Shannon proved that if the rate of the source, as seen by the channel, is less than the channel capacity C, then ''noiseless'' or error free transmission is theoretically possible. The goal of channel encoder is to combat channel noise to achieve a probability of error (Pe)

which approaches zero.

Channel encoders allow a reduction of Pe by selectively reinserting redundancy which has been removed by the source coder. This redundancy allows error detection, and with the proper codes, error correction by the decoder. How close the Pe approaches zero is purely a function of the effort (and money!) spent on channel coding.

A common channel coding technique is the use of ''parity'' bits in digital communication. With parity checking, a bit (or bits) is added to the source word which represents the number (odd or even) of ''ones'' in the word. For example, if odd parity is defined and the source word is 1100101 then a single parity bit of 0 implies an even number of ones (1100101|0).

Parity checking in the example above will _detect_ a _single_ bit error in transmission. A more powerful technique is Hamming codes.

The Hamming code is in a large class of codes known as linear codes. Hamming not only detects errors in transmission, but will _correct_ errors to a certain level. This correction is accomplished by multiplying the received codeword by a matrix known as the _syndrome_. In the case of Hamming codes, the output from this transformation is the bit position in error.

Decoder

As would be expected, the decoder is the inverse

operation which outputs an estimate of the data input to the encoder. The decoder is composed of the channel decoder and the source decoder. The channel decoder uses the redundancy added by the channel encoder to perform error checking and/or correctio . The source decoder takes the ''correct'' data output from the channel decoder and ''decompresses'' the data to produce an estimate of the information source. Information theory has shown that if enough time, complexity, and money is spent on channel encoding, and the information rate is below the channel capacity, then ''error free'' transmission is possible.

Conclusion

This appendix has been a very brief summary of a very large field of study. *The key words underlined* throughout this text are terminology which appear in the theory chapters (2 and 3) of the thesis. For a more thorough, mathematical treatment of the fields of Information Theory and Coding, the reader is encouraged to refer to the textbooks by McEliece (Ref 19) and Berger (Ref 4). The IEEE Press and Benchmark book (Ref 10) is an execellent source for a survey of the key papers in the field of Data Compression.

Appendix C

This appendix contains the 6800 assembly language source programs of the EKG-DAAS.

```
      *:
      *
      * PROGRAM NAME: EKG-EXEC
      * AUTHOR : CAPT. [illegible]
      * VERSION : 1.8
      * VERSION DATE 2 OCT 80
      *
      *
      * PROGRAM DESCRIPTION
      *
      * THIS PROGRAM IS THE EXECUTIVE ROUTINE WHICH
      * CONTROLS THE EXORCISER EKG DATA ACQUISITION
      * SYSTEM. THIS ROUTINE CALLS OVERLAYED PROGRAMS
      * WHICH PERFORM DATA COLLECTION,COMPRESSION,STORAGE
      * AND RECONSTRUCTION.
      * THIS SOFTWARE IS IN SUPPORT OF THESIS RESEARCH
      * TO IDENTIFY THE MOST EFFICIENT EKG DATA COMPRESS-
      * ION ALGORITHM IN THE TEST SET.
      *
      * COMMAND OPTIONS
      *
      * 0=STORAGE WITHOUT COMPRESSION (10 BIT)
      *
      * 1=COMMPRESS WITH ALGOR TOLAN-A
      *
      * 2=COMPRESS WITH ALGOR TOLAN-B
      *
      * 3=COMPRESS WITH ALGOR DOWER
      *
      * 4=COMPRESS WITH ALGOR 2ND ORDER INTERPOL
      *
      * 5=COMPRESS WITH ALGOR TURNPT
      *
      * 6=DISPLAY COLLECTED DATA & STATS
      *
      * 7=JUMP PROGRAM CONTROL TO DOS
      *
      * 8=JUMP PROGRAM CONTROL TO EXBUG
      *
      * 9=LOAD & EXECUTE OTHER OVERLAYS
      *
      * S=SAVE CUR MEM FILE TO DISK
      *
      * START OF PROGRAM
      *
      *
1D00          ORG     $1D00     PROGRAM START LOCATION
      *
              OPT     O         ASSB OPT TO CREATE [illegible] FILE
              OPT     NOG       ASSB OPT TO SUP FCC LIST
      *
      *
      * LABLE DECLARATIONS
      *
```

```
                * SUPPORT SUBROUTINE ADDRESSES
                *
[illegible]
[illegible]
CA36    KEYBD0  EQU     $CA36   EOS. KEYBD INPUT ROUTINE
F000    EXBUG   EQU     $F000   EXBUG. EXBUG ENTRY PT
2800    DOS     EQU     $2800   DOS. DOS ENTRY PT
C75B    WRITE0  EQU     $C75B   EOS. CLR PASSWRD BUFFER
CC87    CLRPAS  EQU     $CC87   EOS. CLR PASSWRD BUFFER
C7C8    DRIVE   EQU     $C7C8   EOSIO. DSK DRIVE SELECT
C803    RLIB    EQU     $C803   EOSIO.
C807    WLIB    EQU     $C807   EOSIO.
CE52    LOAD2   EQU     $CE52   EOS. LOAD PRGM ROUTINE
CBDA    DOSTR2  EQU     $CBDA   EOS. ALT EOS ENTRY LOC
E055    BYTE    EQU     $E055   MIKBUG. GET TO HEX DIG FRM TE
                *
                * DATA BUFFERS
                *
3066    NAME    EQU     $3066   NAME BUFFR FOR FILE I/O
3060    TEMPX1  EQU     $3060   TEMP 2 BYTE STOR BUFFR
3062    TEMPX2  EQU     $3062   TEMP 2 BYTE STOR BUFFR
3058    STARTX  EQU     $3058   LOWEST ADDR USED IN PRGM
305A    ENDX    EQU     $305A   HIGHEST ADDR USED IN PRGM
305C    GOX     EQU     $305C   START EXECUTE ADDR
001B    PROGX   EQU     $001B   DISK I/O ERR VEC ADDR
0008    BMEMH   EQU     $0008
3002    ENDBUF  EQU     $3002   BUF WITH ADDR OF LAST CHAR IN
3400    HDRSTR  EQU     $3400   START OF MEM FILE HDR SECTOR
3004    BUFFER  EQU     $3004   KEYED INPUT BUFFER START
3057    TYPE    EQU     $3057   TYPE OF FILE FOR I/O
1C96    STKSAV  EQU     $1C96   TEMP STACK SAVE BUFFER
1C98    CPRTYP  EQU     $1C98   COMPRESSION TYPE BUFFER
1C9A    COUNT   EQU     $1C9A   GENERAL 8 BIT COUNTER
1C9B    NAMPTR  EQU     $1C9B   NAME POINTER FOR CONSOLE I/O
1C9D    VECSAV  EQU     $1C9D   IRQ VEC SAVE BUFFER
1C9F    OLAYGO  EQU     $1C9F   OVERLAY EXECUTE FLAG
1CA0    LGOFLG  EQU     $1CA0   PRSTAT LOAD VS EXECUTE FLG
1CA1    FILHLC  EQU     $1CA1   FILHDR SUB ADDR PASS BUF
1CA3    SAVELC  EQU     $1CA3   SAVEFL SUB ADDR PASS BUF
1CA5    HXASLC  EQU     $1CA5   HXASC  SUB ADDR PASS BUF
1CA7    HXBUF   EQU     $1CA7   HXASC PARAMATER BUFFER
1CA9    PDFPLC  EQU     $1CA9   PDFPRT SUB ADDR PASS BUF
1CAB    OVRLLC  EQU     $1CAB   OVRLAY SUB ADDR PASS BUF
1CAD    OVRBUF  EQU     $1CAD   OVRLAY PARAMATER PASS BUFFER
3490    LOOPCT  EQU     $3490   TOTL CAL LOOPS EXEC IN UNIT
3494    SAMPNO  EQU     $3494   NUM OF SAMPLES TAKEN
3496    LPCAL   EQU     $3496   NUM OF CAL LOOPS PER 1 [illegible]
3497    MAXZ    EQU     $3497   MAX VLU IN CH Z
3498    MAXZLO  EQU     $3498   LOC OF MAX VLU IN CH Z
349A    MINZ    EQU     $349A   MIN VLU IN CH Z
349B    MINZLO  EQU     $349B   LOC OF MIN VLU IN CH Z
349D    MAXY    EQU     $349D   MAX VLU IN CH Y
349E    MAXYLO  EQU     $349E   LOC OF MAX VLU IN CH Y
34A0    MINY    EQU     $34A0   MIN VLU IN CH Y
```

```
        34A1       MINYLO EQU    $34A1     LOC OF MIN VLU IN CH Y
        34A3       MAXX   EQU    $34A3     MAX VLU IN CH X
        34A4       MAXXLO EQU    $34A4     LOC OF MAX VLU IN CH X
        34A6       MINX   EQU    $34A6     MIN VLU IN CH X
        34A7       MINXLO EQU    $34A7     LOC OF MIN VLU IN CH X
        34A9       MEMBIT EQU    $34A9     NUM OF BITS AVAIL FOR STO
        34AC       DTABIT EQU    $34AC     NUM OF BITS USED TO STO DTA
        34B0       XBITS  EQU    $34B0     NUM OF BITS USED TO STO X
        34B3       YBITS  EQU    $34B3     NUM OF BITS USED TO STO Y
        34B6       ZBITS  EQU    $34B6     NUM OF BITS USED TO STO Z
        34B9       TBITS  EQU    $34B9     NUM OF BITS USED TO STO TIME
        34BC       ACELCT EQU    $34BC     # BITS FED TO VAR LEN CODER
        34C2       BASSAV EQU    $34C2     SAVE LOC FOR VLU   $3620 FRM
        3460       MAXMIN EQU    $3460     START OF MAX,MIN ASCII BUFFER
        34C3       ENTRPY EQU    $34C3     START OF ENTROPY TABLE IN ASC
        3C00       SECZRO EQU    $3C00     SEC 0 OF MEM FILE
        0019       SAVEX  EQU    $0019     TEMP LOC TO SAVE INDEX REG
        3500       XPDF   EQU    $3500     LOC OF X PDF BUFFER
        3700       YPDF   EQU    $3700     LOC OF Y PDF BUFFER
        3900       ZPDF   EQU    $3900     LOC OF Z PDF BUFFER
        3B00       TPDF   EQU    $3B00     LOC OF TIME PDF BUFFER
        8000       BUFEND EQU    $8000     END OF MEM BUF
                   *
                   * HARDWARE ADDRESSES
                   *
        E500       DACZRO EQU    $E500     DAC 0 ADDRESS
        E502       DACONE EQU    $E502     D :THIS ROUTINE IS THE EXECUTIVE
                   * CONTROLLER OF THE EKG DATA ACQ SYS
                   *
                   *
1D00 0F            START  SEI              STOP POSSIBLE INTR ON RESET
1D01 CE 4000              LDX    #$4000
1D04 FF E500              STX    DACZRO    CLR DACS & SET SEL 1 HIGH
1D07 FF E502              STX    DACONE
1D0A CE 214A              LDX    #FILHDR
1D0D FF 1CA1              STX    FILHLC    PUT FILHDR ADDR IN PASS BUFF
1D10 B6 3620              LDA A  $3620
1D13 B7 34C2              STA A  BASSAV    SAVE CURRENT #$3620 IN BUF FO
1D16 CE 208E              LDX    #SAVEFL
1D19 FF 1CA3              STX    SAVELC    PUT SAVEFL ADDR IN PASS BUF
1D1C CE 23DB              LDX    #HXASC
1D1F FF 1CA5              STX    HXASLC    PUT HXASC ADDR IN PASS BUF
1D22 CE 22C2              LDX    #PDFPRT
1D25 FF 1CA9              STX    PDFPLC    PUT PDFPRT ADDR IN PASS BUF
1D28 CE 2067              LDX    #OVRLAY
1D2B FF 1CAB              STX    OVTLIC    PUT OVRLAY ADDR IN PASS BUF
1D2E CE 1D00              LDX    #START
1D31 FF 2801              STX    DOS+1
1D34 7F 1C9F              CLR    OLAYGO    CLR OVERLAY FCN FLAG
1D37 7F 1CA0              CLR    LGOFLG    CLR FRSTAT LOAD FLG
1D3A 4F                   CLR A
1D3B BD C7C8              JSR    DRIVE     INSURE EXEC TLK TO DRIVE 0
1D3E BD 1E6F              JSR    CMDIN     PROMPT & GET IN CMD
1D41 C1 30                CMP B  #'0       IS CMD ASCII 0?
```

```
1D43 26 09               BNE    EXCM1     NO. CHECK OTHER CMDS
1D45 CE 1DC7             LDX    #NOCPR    LOAD OVERLAY NAME PTR
[illegible] FF 1CAD      STX    OVRBUF
[illegible] 7E 2067      JMP    OVRLAY    LOAD & EXECUTE OVRLAY
1D4E C1 31      EXCM1    CMP B  #'1
1D50 26 09               BNE    EXCM2
1D52 CE 1DCF             LDX    #TOLAN1
1D55 FF 1CAD             STX    OVRBUF
1D58 7E 2067             JMP    OVRLAY
1D5B C1 32      EXCM2    CMP B  #'2
1D5D 26 09               BNE    EXCM3
1D5F CE 1DD7             LDX    #TOLAN2
1D62 FF 1CAD             STX    OVRBUF
1D65 7E 2067             JMP    OVRLAY
1D68 C1 33      EXCM3    CMP B  #'3
1D6A 26 09               BNE    EXCM4
1D6C CE 1DDF             LDX    #DOWNR
1D6F FF 1CAD             STX    OVRBUF
1D72 7E 2067             JMP    OVRLAY
1D75 C1 34      EXCM4    CMP B  #'4
1D77 26 09               BNE    EXCM5
1D79 CE 1DE6             LDX    #INTER
1D7C FF 1CAD             STX    OVRBUF
1D7F 7E 2067             JMP    OVRLAY
1D82 C1 35      EXCM5    CMP B  #'5
1D84 26 09               BNE    EXCM6
1D86 CE 1DEE             LDX    #TURNPT
1D89 FF 1CAD             STX    OVRBUF
1D8C 7E 2067             JMP    OVRLAY
1D8F C1 36      EXCM6    CMP B  #'6
1D91 26 09               BNE    EXCM7
1D93 CE 1DF6             LDX    #DISPL
1D96 FF 1CAD             STX    OVRBUF
1D99 7E 2067             JMP    OVRLAY
1D9C C1 37      EXCM7    CMP B  #'7
1D9E 26 09               BNE    EXCM8
1DA0 CE 2BCD             LDX    #$2BCD
1DA3 FF 2801             STX    DOS+1
1DA6 7E 2800             JMP    DOS
1DA9 C1 38      EXCM8    CMP B  #'8
1DAB 26 03               BNE    EXCM9
1DAD 7E F000             JMP    EXBUG
1DB0 C1 39      EXCM9    CMP B  #'9
1DB2 26 09               BNE    EXCMS
1DB4 CE 1DFE             LDX    #OVLMSG
1DB7 BD CA87             JSR    OUTPUT
1DBA 7E CBDA             JMP    DOSTR2
1DBD C1 53      EXCMS    CMP B  #'S
1DBF 26 03               BNE    EXCEND
1DC1 BD 208E             JSR    SAVEFL
1DC4 7E 1D00    EXCEND   JMP    START
                *
1DC7 4E         NOCPR    FCC    /NOCPRS* /
1DCF 54         TOLAN1   FCC    /TOLAN-A*/
```

```
1DD7 54          TOLAN2 FCC    /TOLAN-B*/
1DDF 44          DOWER  FCC    /DOWER* /
[illegible] 49   INTER  FCC    /INTERPL*/
[illegible] 54   TURNPT FCC    /TURNPT* /
1DF6 44          DISPL  FCC    /DISPLAY*/
1DFE 1A          OVLMSG FCB    $1A
1DFF 45                 FCC    /ENTER "RUN FILNAME" TO LOAD & EXECUT
1E25 4F                 FCC    /OVERLAY "FILENAME"./
1E38 0D0A               FDB    $0D0A
1E3A 28                 FCC    /(WARNING: OVERLAY MUST FIT BETWEEN /
1E5D 30                 FCC    /0100-1400)/
1E67 0D0A               FDB    $0D0A,$0D0A,$003F,$2004
                 *
                 * END EKG-EXEC
                 *
                 *FUNCTION :CMDIN
                 *INPUTS (REG) :NONE
                 *OUTPUTS (REG):B
                 *CALLS :OUTPUT,KEYBD0
                 *DESTROYS :ALL REGS
                 *PURPOSE :TO PRINT PROMPT MESSAGE AND READ COMMAND
                 * FROM CONSOLE.
                 *
1E6F CE 1E7F     CMDIN  LDX    #CMDMSG  LOAD OUT PTR
1E72 BD CA8F            JSR    OUTNCR
1E75 BD CA36            JSR    KEYBD0
1E78 FE 3002            LDX    ENDBUF   SET PRT TO LAST CHAR ENTERED
1E7B 09                 DEX
1E7C E6 00              LDA B  0,X
1E7E 39                 RTS
                 *
1E7F 1A07        CMDMSG FDB    $1A07,$0D0A,$0A0A,$0A0A,$0A0A
1E89 45                 FCC    /EKG DATA ACQUISITION SYSTEM/
1EA4 0D0A               FDB    $0D0A,$0D0A
1EA8 43                 FCC    /COMMAND OPTIONS:/
1EB8 0D0A               FDB    $0D0A,$0D0A
1EBC 44                 FCC    /DATA ACQUISITION/
1ECC 0D0A               FDB    $0D0A,$0D0A
1ED0 20                 FCC    / 0=STORAGE WITH NO COMPRESSION/
1EEE 0D0A               FDB    $0D0A
1EF0 20                 FCC    / 1=COMPRESSION VIA TOLAN ALGORITHM A
1F14 0D0A               FDB    $0D0A
1F16 20                 FCC    / 2=COMPRESSION VIA TOLAN ALGORITHM B
1F3A 0D0A               FDB    $0D0A
1F3C 20                 FCC    / 3=COMPRESSION VIA DOWER ALGORITHM/
1F5E 0D0A               FDB    $0D0A
1F60 20                 FCC    / 4=COMPRESSION VIA 2ND ORDER INTERPL
1F89 0D0A               FDB    $0D0A
1F8B 20                 FCC    / 5=COMPRESSION VIA TURNING POINT ALG
1FB1 0D0A               FDB    $0D0A,$0D0A
1FB5 44                 FCC    /DATA DISPLAY/
1FC1 0D0A               FDB    $0D0A,$0D0A
1FC5 20                 FCC    / 6=PRINT DATA STATISTICS OR /
1FE1 44                 FCC    /DISPLAY DATA/
```

```
1FED 0D0A          FDB    $0D0A,$0D0A
1FF1 50            FCC    /PROGRAM CONTROL/
2000 0D0A          FDB    $0D0A,$0D0A
2004 20            FCC    / 7=RETURN TO DOS/
2014 0D0A          FDB    $0D0A
2016 20            FCC    / 8=RETURN TO EXBUG/
2028 0D0A          FDB    $0D0A
202A 20            FCC    / 9=LOAD & EXECUTE USER ENTERED OVERL
2050 0D0A          FDB    $0D0A,$0D0A
2054 45            FCC    /ENTER COMMAND NOW=/
2066 04            FCB    4
                *
                * END CMDIN
                *
                *
                *FUNCTION :OVRLAY
                *INPUTS (REG) :X
                *OUTPUTS (REG) :NONE
                *CALLS :CLRNAM,PUTNAM,OVERLAY AT 0,X
                *DESTROYS :ALL REGISTERS
                *PURPOSE :TO LOAD AND THEN EXECUTE OVERLAYS
                * ASSOCIATED WITH EKG-EXEC &
                * SPECIFIED BY THE INPUT COMMAND.
                *
                *
                *
2067 BD 20E1    OVRLAY JSR    CLRNAM    CLR NAME BUFFER
206A BD CC87           JSR    CLRPAS    CLEAR PASSWORD BUFFER
206D CE 3066           LDX    #NAME     PUT DOS FILENAME BUF IN NAMPT
2070 FF 1C9B           STX    NAMPTR
2073 FE 1CAD           LDX    OVRBUF    POINT X TO OVRLAY NAME
2076 BD 20EF           JSR    PUTNAM    PUT OVRLAY NAME IN NAME
2079 86 22             LDA A  #$22      DEFINE FILE TYPE
207B B7 3057           STA A  TYPE
207E BD CE52           JSR    LOAD2     LOAD FILE POINTED TO BY STK
2081 86 AA             LDA A  #$AA      SET UP COMPARE FOR OVRLAY FCN
2083 B1 1C9F           CMP A  OLAYGO    IS FLAG TRUE?
2086 27 05             BEQ    OVRRTS    YES. TREAT OVRLAY AS SUBR
2088 8E A049           LDS    #$A049    NO. TREAT OVRLAY AS ABS JUMP
208B 6E 00             JMP    0,X       JUMP TO OVERLAY
208D 39         OVRRTS RTS
                *
                * END OVRLAY
                *
                *
                *FUNCTION :SAVEFL
                *INPUTS (REG) :NONE
                *OUTPUTS (REG) :NONE
                *CALLS :CLRPAS,DRIVE,WLIB,WRITE0,OUTPUT
                *DESTROYS :ALL REGISTERS
                *PURPOSE :TO SAVE EKG DATA FILES RESIDING IN
                *IN MEMORY LOCATIONS 3A00-7FFF ON DISK. FILE NAME IS
                * PICKED OFF OF FILE HEADER AT LOC 3A0[illegible]-3A0[illegible].
                *
```

```
                    *
208E 8D 51   SAVEFL BSR    CLRNAM    CLR NAME BUFFER
[illegible]
2093 8D 4F          BSR    CLRNA0    [illegible]
2095 B6 3620        LDA A  $3620     GET ITH LOC FOR SAVE FROM PAS
2098 B7 34C2        STA A  BASSAV    SAVE IN BUFFER
209B CE 2141        LDX    #NAMMSG   SET UP PTR TO PRT NAME
209E FF 1C9B        STX    NAMPTR
20A1 CE 3402        LDX    #HDRSTR+2  POINT X TO FILENM IN DATA
20A4 8D 49          BSR    PUTNAM    PUT FILENAM IN "NAME" BUF
20A6 86 22          LDA A  #$22      SET TYPE OF FILE
20A8 B7 3057        STA A  TYPE
20AB CE 3400        LDX    #HDRSTR   LOAD BUFFER START
20AE FF 3058        STX    STARTX
20B1 CE 7FFF        LDX    #BUFEND-1  LOAD BUFFER END
20B4 FF 305A        STX    ENDX
20B7 CE 2800        LDX    #$2800    LOAD LGO ADDRESS
20BA FF 305C        STX    GOX
20BD DF 1B          STX    PROGX
20BF BD CC87        JSR    CLRPAS    CLR PASSWRD BUFFER
20C2 CE 2119        LDX    #DRVMSG
20C5 BD CA8F        JSR    OUTNCR
20C8 BD E055        JSR    BYTE
20CB BD C7C8        JSR    DRIVE     SELECT DRIVE 01 FOR STORE
20CE BD C807        JSR    WLIB
20D1 DE 08          LDX    ENEMH
20D3 BD C75B        JSR    WRITE0    WRITE BUFFER TO DISK
20D6 4F             CLR A
20D7 BD C7C8        JSR    DRIVE     RESET DRIVE BACK TO 0
20DA CE 2123        LDX    #CATMSG   LOAD OUTPUT POINTER
20DD BD CA87        JSR    OUTPUT
20E0 39             RTS
                    *
20E1 CE 3066 CLRNAM LDX    #NAME     POINT TO NAME BUFFER
20E4 C6 08   CLRNA0 LDA B  #$8       SET UP CHAR COUNT
20E6 86 20          LDA A  #$20      ASCII SPACE
20E8 A7 00   CLRNA1 STA A  0,X       FILL NAME BUF WITH SPACES
20EA 08             INX
20EB 5A             DEC B
20EC 26 FA          BNE    CLRNA1
20EE 39             RTS
                    *
20EF BF 1C96 PUTNAM STS    STKSAV    SAVE STACK POINTER
20F2 7F 1C9A        CLR    COUNT     CLEAR COUNTER
20F5 35             TXS              TRANSFER X TO STACK
20F6 CE 3066        LDX    #NAME     PICK UP DOS [illegible]
20F9 C6 08          LDA B  #8        SET UP NAME LENGTH [illegible]
20FB 32      PUTNA1 PUL A            PULL CHAR FROM STACK
20FC 81 04          CMP A  #$04      IS IT 4?
20FE 27 15          BEQ    PUTNA2    YES. NAME ENDED
2100 A7 00          STA A  0,X       NO. STORE CHAR IN "[illegible]"
2102 08             INX
2103 FF 3060        STX    TEMPX1    SAVE INDEX REG
2106 FE 1C9B        LDX    NAMPTR    LOAD CURRENT [illegible]
```

```
2109 A7 00                STA A  0,X       STORE IN NAMEMS
210B 08                   INX              INCR POINTER
210C FF 1C9B              STX    NAMPTR
210F FE 3060              LDX    TEMPX1    RESTORE INDEX REG
2112 5A                   DEC B
2113 26 E6                BNE    PUTNA1    8 CHARS YET?
2115 FE 1C96  PUTNA2 LDS         STKSAV    YES. RESTORE STACK
2118 39                   RTS
                     *
2119 0D0A     DRVMSG FDB         $0D0A
211B 44                   FCC    /DRIVE? /
2122 04                   FCB    4
2123 1A07     CATMSG FDB         $1A07,$0D0A,$0D0A
2129 44                   FCC    /DATA SAVED ON DISK FILE /
2141 0008     NAMMSG RMB         8
2149 04                   FCB    4
                     *
                     * END SAVEFL
                     *
                     *
                     *FUNCTION :FILHDR
                     *INPUTS (REG) :NONE
                     *OUTPUTS (REG) :NONE
                     *CALLS :OUTPUT,KEYBD0
                     *DESTROYS :ALL REGISTERS
                     *PURPOSE :TO INIT DATA BUFFER WITH ACCOUNT
                     * DATA (NAME,DATE,TIME,ETC) AND CLEAR FOR NEW
                     * DATA STORAGE.
                     *
214A BD 21D7 FILHDR JSR    CLRBUF   FILL DATA BUF WITH NULS
214D BD 21C9        JSR    CLRSEC   FILL MEM FILE SEC 3400 WITH A
2150 BD 2288        JSR    PDFCLR   INITIALIZE COLL PARAM
2153 FE 1C98        LDX    CPRTYP   PICK UP COMPR TYPE
2156 FF 3400        STX    HDRSTR   STR IN HEADER SECTOR
2159 CE 21E5        LDX    #NAMEMS  OUT PTR
215C BD CA8F        JSR    OUTNCR   "NAME FILE PLEASE?"
215F BD CA36        JSR    KEYBD0
2162 C6 02          LDA B  #$02     HEADER STOR OFFSET
2164 8D 42          BSR    TXTSTR   STORE FILNAME IN HEADER SEC
2166 37             PSH B           SAVE NEXT AVAIL HEAD LOC
2167 CE 2201        LDX    #SUBJMS
216A BD CA8F        JSR    OUTNCR   "TEST SUBJECT ID?"
216D BD CA36        JSR    KEYBD0
2170 33             PUL B           RETRIEVE STOR OFFSET
2171 8D 35          BSR    TXTSTR
2173 37             PSH B
2174 CE 221F        LDX    #RATEMS
2177 BD CA8F        JSR    OUTNCR   "COLLECTION SAMPLING RATE?"
217A BD CA36        JSR    KEYBD0
217D 33             PUL B
217E 8D 28          BSR    TXTSTR
2180 37             PSH B
2181 CE 2247        LDX    #DATEMS
2184 BD CA8F        JSR    OUTNCR   "DATE (05 JUL 80)?"
```

```
[illegible] 2187 BD CA36          JSR    KEYBD0
2188A 33                          PUL B
[illegible] 8D 1B                 BSR    TXTSTR
[illegible] 37                    PSH B
218E CE 225E                      LDX    #TIMEMS
2191 BD CA8F                      JSR    OUTNCR   "TIME (1420)?"
2194 BD CA36                      JSR    KEYBD0
2197 33                           PUL B
2198 8D 0E                        BSR    TXTSTR
219A 37                           PSH B
219B CE 2270                      LDX    #CONTMS
219E BD CA8F                      JSR    OUTNCR   "COMMENTS (MAX 80)?"
21A1 BD CA36                      JSR    KEYBD0
21A4 33                           PUL B
21A5 8D 01                        BSR    TXTSTR
21A7 39                           RTS
                     *
21A8 BF 1C96 TXTSTR  STS    STKSAV    SAVE STACK
21AB 8E 3003         LDS    #BUFFER-1 POINT TO INPUT BUFFER
21AE CE 3400         LDX    #HDRSTR   POINT TO HEADER
21B1 4F              CLR A            SET A TO ZERO
21B2 4C      TXTST1  INC A
21B3 08              INX
21B4 11              CBA              IS A = OFFSET?
21B5 2D FB           BLT    TXTST1    NO. KEEP INCR'TING
21B7 32      TXTST2  PUL A            YES. GET CHAR FROM STACK
21B8 81 04           CMP A  #$04      IS IT 4
21BA 27 06           BEQ    TXTRTN    YES. END TEXT ,RETURN
21BC A7 00           STA A  0,X       NO. STORE CHAR IN HEADER
21BE 5C              INC B
21BF 08              INX
21C0 20 F5           BRA    TXTST2
21C2 A7 00   TXTRTN  STA A  0,X
21C4 5C              INC B
21C5 BE 1C96         LDS    STKSAV    RESTORE STACK
21C8 39              RTS
21C9 CE 3400 CLRSEC  LDX    #HDRSTR   POINT TO DATA BUF START
21CC 86 20           LDA A  #$20      ASCII SPACE
21CE A7 00   CLRSE1  STA A  0,X       STORE SPACE
21D0 08              INX
21D1 8C 3500         CPX    #XPDF     IS SECTOR SPACED?
21D4 26 F8           BNE    CLRSE1    COUNT > ZRO?
21D6 39              RTS              NO. RTN
                     *
21D7 CE 3400 CLRBUF  LDX    #HDRSTR   POINT TO DATA BUF
21DA 86 00           LDA A  #00       ASCII NULL
21DC A7 00   CLRBU1  STA A  0,X       STORE NULL
21DE 08              INX
21DF 8C 8000         CPX    #BUFEND   IS INDEX REG AT END?
21E2 26 F8           BNE    CLRBU1    NO. KEEP GOING
21E4 39              RTS              YES. RETURN
                     *
21E5 0D0A    NAMEMS  FDB    $0D0A,$0D0A
21E9 4E              FCC    /NAME DATA FILE PLEASE? /
```

```
2200 04                 FCB    4
2201 0D0A      SUBJMS   FDB    $0D0A,$0D0A
[illegible]                    [illegible] R? /
221E 04                 FCB    4
221F 0D0A      RATEMS   FDB    $0D0A,$0D0A
2223 43                 FCC    /COLLECTION SAMPLING RATE (500 HZ)? /
2246 04                 FCB    4
2247 0D0A      DATEMS   FDB    $0D0A,$0D0A
224B 44                 FCC    /DATE (05 JUL 80)? /
225D 04                 FCB    4
225E 0D0A      TIMEMS   FDB    $0D0A,$0D0A
2262 54                 FCC    /TIME (1423)? /
226F 04                 FCB    4
2270 0D0A      CONTMS   FDB    $0D0A,$0D0A
2274 43                 FCC    /COMMENTS (MAX=80)? /
2287 04                 FCB    4
               *
               * END FILHDR
               *
               *FUNCTION :PDFCLR
               *INPUTS (REG) :NONE
               *OUTPUTS (REG):NONE
               *CALLS :NOTHING
               *DESTROYS :A,B,X,CC
               *PURPOSE :TO INITIALIZE STATISTIC VAR & CLEAR
               * PDF BUFFERS
               *
2288 4F        PDFCLR   CLR A                 FILL PARAM BUFF WITH 0
2289 CE 3490            LDX    #LOOPCT
228C A7 00     PDFCL0   STA A  0,X
228E 08                 INX
228F 8C 34C3            CPX    #ENTRPY        FILL WITH 0 UP TO ENTROPY BUF
2292 26 F8              BNE    PDFCL0
2294 CE 021F            LDX    #$021F         STORE MAX POSSIBLE BITS IN ME
2297 86 F0              LDA A  #$F0
2299 FF 34A9            STX    MEMBIT
229C B7 34AB            STA A  MEMBIT+2
229F 4F                 CLR A
22A0 CE 3500            LDX    #XPDF          NOW CLR PDF BUF AREA
22A3 A7 00     PDFCL1   STA A  0,X
22A5 08                 INX
22A6 8C 3C00            CPX    #SECZRO
22A9 26 F8              BNE    PDFCL1
22AB 86 80              LDA A  #$80           NOW SET UP INITIAL MAX & MINS
22AD C6 7F              LDA B  #$7F
22AF B7 3497            STA A  MAXZ
22B2 B7 349D            STA A  MAXY
22B5 B7 34A3            STA A  MAXX
22B8 F7 349A            STA B  MINZ
22BB F7 34A0            STA B  MINY
22BE F7 34A6            STA B  MINX
22C1 39                 RTS
               *
               * END PDFCLR
```

```
2327 DE 19                 LDX    SAVEX      GET CUR PDF ADDR
2329 A6 00                 LDA A  0,X        LDA MSB BYTE OF PDF VLU
232B E6 01                 LDA B  1,X        LD LSB
232D CE 23CD               LDX    #PDFMS5
2330 FF 1CA7               STX    HXBUF
2333 BD 23DB               JSR    HXASC      CONV & STO MSB IN ASC STR
2336 17                    TBA
2337 CE 23CF               LDX    #PDFMS5+2
233A FF 1CA7               STX    HXBUF
233D BD 23DB               JSR    HXASC      CONV & STORE LSB IN ASC STR
2340 CE 23BB               LDX    #PDFMS3
2343 BD CA87               JSR    OUTPUT     PRINT STRING
2346 32                    PUL A             RETRIEV VLU INDEX
2347 DE 19                 LDX    SAVEX      RETRIEC CUR PDF ADDR
2349 39                    RTS
                   *
234A 1A0C          PDFMS1  FDB    $1A0C,$0D0A,$0A0A,$0A0A,$0A0A
2354 3A                    FCC    /:FILE /
235A 0008          PDFNAM  RMB    8
2362 0D0A                  FDB    $0D0A,$0D0A
2366 43                    FCC    /CHANNEL /
236E 0001          PDFMS2  RMB    1
236F 20                    FCC    / AMPLITUDE DISTRIBUTION/
2386 0D0A                  FDB    $0D0A,$0D0A
238A 44                    FCC    /DATA VALUE/
2394 20                    FCC    /               NUMBER OF OCCURENCES/
23B6 0D0A                  FDB    $0D0A,$0D0A
23BA 04                    FCB    4
23BB 20            PDFMS3  FCC    /    /
23BF 0002          PDFMS4  RMB    2
23C1 2E                    FCC    /............/
23CD 0004          PDFMS5  RMB    4
23D1 20                    FCC    / (HEX)/
23D7 04                    FCB    4
23D8 0D23          PDFMS6  FDB    $0D23
23DA 04                    FCB    4
                   *
                   *
                   * END PDFPRT
                   *
                   *FUNCTION :HXASC
                   *INPUTS (REG) :A,X,DTA FOR CONV,LOC OF ASC STOR
                   *OUTPUTS(REG) :NONE
                   *CALLS :NOTHING
                   *DESTROYS :A,CC
                   *PURPOSE :TO CONVERT 1 BYTE OF HEX DATA
                   * INTO 2 BYTES OF ASCII
                   *
                   *
23DB FE 1CA7       HXASC   LDX    HXBUF
23DE 36                    PSH A             SAVE DTA ON STK
23DF 44                    LSR A             SHIFT TOP 4 BITS TO LOW 4
23E0 44                    LSR A
23E1 44                    LSR A
```

```
                  *
                  *
                  *FUNCTION :PDFPRT
                  *INPUTS (REG):NONE
                  *OUTPUTS (REG):NONE
                  *CALLS :OUTPUT
                  *DESTROYS(REG):A,B,X,CC
                  *PURPOSE :TO PRINT AMPLITUDE DISTRIBUTION
                  * DATA COMPILED DURING DATA COLLECTION.
                  *
22C2 CE 235A PDFPRT LDX     #PDFNAM   PUT CUR MEM FILE NAME IN MSG
22C5 FF 1C9B        STX     NAMPTR
22C8 CE 3402        LDX     #HDRSTR+2
22CB BD 20EF        JSR     PUTNAM
22CE CE 234A        LDX     #PDFMS1   LOAD MSG POINTER
22D1 86 58          LDA A   #'X
22D3 B7 236E        STA A   PDFMS2    STOR ASCII X TO MSG
22D6 BD CA87        JSR     OUTPUT    "CHANNEL X AMPLITUDE .."
22D9 CE 3500        LDX     #XPDF     POINT TO X PDF DTA
22DC 86 7F          LDA A   #$7F      INIT COUNTER
22DE 8D 25          BSR     PDFPR1    PRINT X PDF DATA
22E0 CE 234A        LDX     #PDFMS1   STOR ASCII Y TO MSG
22E3 86 59          LDA A   #'Y
22E5 B7 236E        STA A   PDFMS2
22E8 BD CA87        JSR     OUTPUT
22EB CE 3700        LDX     #YPDF     POINT TO Y PDF DTA
22EE 86 7F          LDA A   #$7F
22F0 8D 13          BSR     PDFPR1
22F2 CE 234A        LDX     #PDFMS1
22F5 86 5A          LDA A   #'Z
22F7 B7 236E        STA A   PDFMS2
22FA BD CA87        JSR     OUTPUT
22FD CE 3900        LDX     #ZPDF
2300 86 7F          LDA A   #$7F
2302 8D 01          BSR     PDFPR1
2304 39             RTS
                  *
2305 4C      PDFPR1 INC A
2306 81 7F          CMP A   #$7F      256 VLUES PRNT'D YET?
2308 27 07          BEQ     PDFPR2    YES. EXIT
230A BD 231B        JSR     PRTVLU    NO. PRINT VLUE TO CONSOLE
230D 08             INX
230E 08             INX
230F 20 F4          BRA     PDFPR1    PRINT NEXT VLU
2311 BD 231B PDFPR2 JSR     PRTVLU
2314 CE 23D8        LDX     #PDFMS6
2317 BD CA8F        JSR     OUTNCR
231A 39             RTS
                  *
231B 36      PRTVLU PSH A             SAVE VALUE INDEX IN A
231C DF 19          STX     SAVEX     SAVE LOC POINTED TO BY X
231E CE 23BF        LDX     #PDFMS4   PICK UP STRING PTR
2321 FF 1CA7        STX     HXBUF
2324 BD 23DB        JSR     HXASC     CONV VLU INDEX TO ASCII & STR
```

```
23E2 44                    LSR A
23E3 8D 08                 BSR    HXTOAS    CONV SHFT'D BITS TO ASC
23E5 A7 00                 STA A  0,X       STR IN ALPH STRING
23E7 32                    PUL A
23E8 8D 03                 BSR    HXTOAS    CONV LOW 4 TO ASC
23EA A7 01                 STA A  1,X       STR IN ALPH STRING
23EC 39                    RTS
                   *
23ED 84 0F         HXTOAS  AND A  #$0F      CLR TOP 4 BITS
23EF 81 0A                 CMP A  #$0A      IS NUM GE HEX A?
23F1 2C 03                 BGE    HXTOA1    YES. BRANCH & ADD BIAS OF 37
23F3 8B 30                 ADD A  #$30      NO. ADD BIAS OF30 & RET
23F5 39                    RTS
23F6 8B 37         HXTOA1  ADD A  #$37
23F8 39                    RTS
                   *
                   * END HXASC
                   *
                   * END OF EKG-EXEC ROUTINES
                   *
                           END
```

```
**************************************************
*
*     OVERLAY NAME        :DISPLAY
*     AUTHOR              :CAPT [illegible]
*     VERSION              :1.5
*     VERSION DATE         :4 NOV 80
*
**************************************************
*
*                PROGRAM DESCRIPTION
*
*     THIS PROGRAM DISPLAYS THE STATISTICAL DATA
* CALCULATED DURING THE COLLECTION OF AN EKG
* RECORDING.  THE USER HAS SEVERAL MODES OF DATA
* OUTPUT AND THEY ARE LISTED IN THE COMMAND STRING
* BELOW.  THIS PROGRAM ALSO PRODUCES PROBABILITY
* DENSITY FUNCTION (PDF) PLOTS OF THE DATA COLLECTED
* ON CHANNEL X,Y,OR Z VIA D/A CONVERTERS. THE DATA
* (PDF) IS DISPLAYED ON AND OSCILLISCOPE.
* IN ADDITION, THE COMMAND STRING ALLOWS RECONSTRUCT
* AND DISPLAY OF PREVIOUSLY COLLECTED DATA ON THE
* OSCILLISCOPE.  THE RECONSTRUCTION ROUTINES ARE
* OVERLAYED OVER THE PRINT STATISTICS (PRSTAT)
* ROUTINES.
*
*   COMMAND OPTIONS
*
*       0= RETURN TO EKG-EXEC
*       1= PRINT CURRENT MEMORY FILE STATISTICS
*       2= PRINT PDF AMPLITUDE TABLES
*       3= DISPLAY XPDF ON THE OSCILLISCOPE
*       4= DISPLAY YPDF ON THE OSCILLISCOPE
*       5= DISPLAY ZPDF ON THE OSCILLISCOPE
*       6= RECONSTRUCT & DISPLAY CH X ON
*          THE OSCILLISCOPE
*       7= RECONSTRUCT & DISPLAY CH Y ON
*          THE OSCILLISCOPE
*       8= RECONSTRUCT & DISPLAY CH Z ON
*          THE OSCILLISCOPE
*       9= LOAD MEMORY FILE FROM DISK
*
**************************************************
**************************************************
*
*            START OF DISPL
*
**************************************************
*
0100        ORG     $0100     PROGRAM ORIGIN
*
            OPT     O         ASSB OPT. LIST ASSEMBLY
            OPT     NOG       ASSB OPT. SUPPRESS FCC LIST
*
**************************************************
```

```
                *
                *          LABEL DECSRIPTIONS
                *
                * [illegible]
                *
        CA8F    OUTNCR EQU     $CA8F    EPROMDOS. OUTPUT STRING WITH
        CA36    KEYBD0 EQU     $CA36    EPROMDOS. INPUT ALPH STRING
        1D00    START  EQU     $1D00    EKG-EXEC.  START ADDRESS
        CCA0    MAKNAA EQU     $CCA0    EPROMDOS. GET FILE NAME FROM
        C803    RLIB   EQU     $C803    EPROMDOS. SET UP READ I/O
        2800    DOS    EQU     $2800    DOS. RESTART LOC
        C43C    READ0  EQU     $C43C    EPROMDOS. READ DISK ROUTINE
        E055    BYTE   EQU     $E055    MIKBUG. INPUT 2 HEX DIG FRM T
        C7C8    DRIVE  EQU     $C7C8    EPROMDOSIO. READ DISK ROUTINE
        0500    PRSTAT EQU     $0500    PRINT COLLEC STAT ROUTINE
                *
                * DATA BUFFERS
                *
        0020    CHNLBF EQU     $0020    CHANNEL DESIGNATOR FOR DECFRS
        3400    HDRSTR EQU     $3400    MEM FILE HDR ADDR
        3500    XPDF   EQU     $3500    LOC OF CH X PDF BUFFER
        3700    YPDF   EQU     $3700    LOC OF CH Y PDF BUFFER
        3900    ZPDF   EQU     $3900    LOC OF CH Z PDF BUFFER
        FFF8    IRQVEC EQU     $FFF8    LOC OF INT VECTOR ADDRESS
        1C9D    VECSAV EQU     $1C9D    TEMP IRQVEC SAVE LOC
        3002    ENDBUF EQU     $3002    BUF WITH ADDR OF LAST CHAR IN
        1C96    STKSAV EQU     $1C96    TEMP STACK STORAGE BUFFER
        0019    SAVEX  EQU     $0019    TEMP INDEX REG STORAGE BUFFER
        000A    EMEMH  EQU     $000A    END MEM ADDR FOR DISK I/O
        001B    PROGX  EQU     $001B    ERR VEC ADDR
        1C9F    OLAYCO EQU     $1C9F    OVRLAY MODE FLAG
        3057    TYPE   EQU     $3057    FILE TYPE BUFFER
        1CA9    PDFPLC EQU     $1CA9    PDFPRT ADDR PASS BUFFER
        1CAB    OVRLLC EQU     $1CAB    OVRLAY ADDR PASS BUFFER
        1CAD    OVRBUF EQU     $1CAD    OVRLAY PARAMETER BUFFER
                *
                * HARDWARE ADDRESSES
                *
        E500    DACZRO EQU     $E500    D/A CONVERTER ZERO
        E400    ADCZRO EQU     $E400    A/D CONVERTER ZERO
                *
                *
                *FUNCTION      :DISPL
                *INPUTS (REG)  :NONE
                *OUTPUTS(REG)  :NONE
                *CALLS         :OUTNCR,KEYBD0
                *DESTROYS      :A,B,CC,X
                *PURPOSE       :DISPL IS THE COMMAND EXEC [illegible]
                *               COMMAND INPUT FROM THE [illegible] AND T
                *               DIRECTING EXECUTION OF THE [illegible] O
                *               IONS.
                *
                *
0100 CE 0100    DISPL  LDX     #DISPL
```

```
0103 FF 2801          STX    DOS+1
0106 CE 025B          LDX    #DISPMS
[illegible]           JSR    [illegible]  "[illegible] ...."
010C BD CA36          JSR    [illegible]  [illegible]
010F FE 3002          LDX    ENDBUF   SET UP IN DATA BUF POINTER
0112 09               DEX
0113 E6 00            LDA B  0,X      GET LAST CHAR INPUT FROM TERM
0115 C1 30            CMP B  #'0      WAS IT 0?
0117 26 03            BNE    DISPL1   NO. CHEK IF 1
0119 7E 1D00          JMP    START    YES. RET TO EKG-EXEC
011C C1 31    DISPL1  CMP B  #'1      WAS COMMAND 1?
011E 26 0C            BNE    DISPL2   NO. CHEKC IF 2
0120 CE 01E4          LDX    #PRSNAM  GET "PRSTAT*" TO SEND TO OVRL
0123 FF 1CAD          STX    OVRBUF   STORE "PRSTAT" TO OVRLAY BUFF
0126 7F 1C9F          CLR    OLAYGO   RESTE OLAYGO FLAG
0129 7E 04D6          JMP    OVRLAY
012C C1 32    DISPL2  CMP B  #'2      WAS COMMAND 2?
012E 26 0F            BNE    DISPL3   NO. CHEK IF 3
0130 BD 04DB          JSR    PDFPRT   PRINT PDF AMPLITUDE TABLES
0133 CE 0463          LDX    #DISMS1
0136 BD CA8F          JSR    OUTNCR   "PRESS RETURN"
0139 BD CA36          JSR    KEYBD0
013C 7E 0100          JMP    DISPL
013F C1 33    DISPL3  CMP B  #'3      WAS COMMAND 3?
0141 26 0F            BNE    DISPL4   NO. CHECK 4
0143 CE 3500          LDX    #XPDF    GET ADDRESS OF X PDF DATA
0146 BD 0470          JSR    MAXSCN   SCAN PDF & FIND SHIFT NECESSA
              *                       TO ALLOW OUTPUT OF PDF VIA 12
0149 BD 04A0          JSR    PDFTRN   TRANSFER X PDF TO WORK BUFFER
014C BD 04BA          JSR    SCALE    SCALE WORK BUFFER BY SHIFT FR
014F 7E 01F4          JMP    PDFOUT   NOW DISPLAY X PDF TO OSCOPE.
0152 C1 34    DISPL4  CMP B  #'4      WAS COMMAND 4?
0154 26 0F            BNE    DISPL5   NO. CHECK 5
0156 CE 3700          LDX    #YPDF    GET ADDRESS OF Y PDF DATA
0159 BD 0470          JSR    MAXSCN
015C BD 04A0          JSR    PDFTRN
015F BD 04BA          JSR    SCALE
0162 7E 01F4          JMP    PDFOUT
0165 C1 35    DISPL5  CMP B  #'5      WAS COMMAND 5?
0167 26 0F            BNE    DISPL6   NO. RET & PRINT COMMAND PROMP
0169 CE 3900          LDX    #ZPDF
016C BD 0470          JSR    MAXSCN
016F BD 04A0          JSR    PDFTRN
0172 BD 04BA          JSR    SCALE
0175 7E 01F4          JMP    PDFOUT
0178 C1 36    DISPL6  CMP B  #'6
017A 26 0F            BNE    DISPL7
017C 7F 0020          CLR    CHNLBF   SET CHNL DESIGNATOR FOR CH X
017F 7F 1C9F          CLR    OLAYGO
0182 CE 01EC          LDX    #DECPRS  LOAD & RUN DECPRS ROUTINE
0185 FF 1CAD          STX    OVRBUF   STORE "DECPRS" TO OVRLAY BUFF
0188 7E 04D6          JMP    OVRLAY
018B C1 37    DISPL7  CMP B  #'7
018D 26 10            BNE    DISPL8
```

```
01621 018F 86 01            LDA A  #1        SET CHNL DESIGNATOR FOR CH Y
01622 0191 97 20            STA A  CHNLBF
01623 0193 7F 1C9F          CLR    OLAYGO
[illegible]
01650 0199 FF 1CAD          STX    OVRBUF    STORE "DECPRS" TO OVRLAY BUFF
01660 019C 7E 04D6          JMP    OVRLAY
01670 019F C1 38    DISPL8  CMP B  #'8
01680 01A1 26 10            BNE    DISPL9
01681 01A3 86 02            LDA A  #2        SET CHNL DESIGNATOR FOR CH Z
01682 01A5 97 20            STA A  CHNLBF
01690 01A7 7F 1C9F          CLR    OLAYGO
01700 01AA CE 01EC          LDX    #DECPRS   LOAD & RUN DECPRS ROUTINE
01710 01AD FF 1CAD          STX    OVRBUF    STORE "DECPRS" TO OVRLAY BUFF
01720 01B0 7E 04D6          JMP    OVRLAY
01730 01B3 C1 39    DISPL9  CMP B  #'9
01740 01B5 26 2A            BNE    CMDERR
01750 01B7 86 22            LDA A  #$22      SET UP FILE TYPE FOR LOAD
01760 01B9 B7 3057          STA A  TYPE
01770 01BC CE 024C          LDX    #NAMMSG   "ENTER FILENAME?"
01780 01BF BD CA8F          JSR    OUTNCR
01790 01C2 BD CCA0          JSR    MAKNAA
01800 01C5 BD C803          JSR    RLIB
01810 01C8 CE FFFF          LDX    #$FFFF
01820 01CB DF 0A            STX    EMEMH
01830 01CD CE 2800          LDX    #DOS
01840 01D0 DF 1B            STX    PROGX
01850 01D2 CE 3400          LDX    #HDRSTR   PUT ADDR TO LOAD IN INDEX
01860 01D5 BD C43C          JSR    READ0
01870 01D8 CE 0456          LDX    #FILEMS
01880 01DB BD CA8F          JSR    OUTNCR
01890 01DE BD CA36          JSR    KEYBD0
01900 01E1 7E 0100  CMDERR  JMP    DISPL
01910               *
01920 01E4 50       PRSNAM  FCC    /PRSTAT* /
01930 01EC 44       DECPRS  FCC    /DECPRS* /
01960               *
01970 01F4 86 AA    PDFOUT  LDA A  #$AA      SET INTERRUPT DONE TEST FLAG
01980 01F6 B7 0248          STA A  DONTST
01990 01F9 0E       PDFOU1  CLI              CLEAR INTERRUPT MASK FOR STOP
02000 01FA 7F 024B          CLR    KOUNT     CLR BUFFER PRT COUNTER
02010 01FD FE FFF8          LDX    IRQVEC    GET CUR IRQ VECTOR
02020 0200 FF 1C9D          STX    VECSAV    SAVE IN TEMP BUFFER
02030 0203 CE 0240          LDX    #STOP     LOAD ADDRESS OF STOPING INTR
02040 0206 FF FFF8          STX    IRQVEC
02050 0209 CE 07FF          LDX    #$07FF    SEND MAX POS PUSLE FOR SCOPE
02060 020C FF E500          STX    DACZRO
02070 020F CE 3200          LDX    #$3200    GET START ADDR OF FOUR BUFFER
02080 0212 7C 024B  PDFOU2  INC    KOUNT     BUMP UP PTR COUNTER
02090 0215 27 0E            BEQ    PDFOU3    HAS WHOLE BUFFER BEEN OUTPUT
02100 0217 A6 00            LDA A  0,X       NO. GET MSB BYTE OF DATA AND D
02110 0219 E6 01            LDA B  1,X       GET LSB
02120 021B B7 E500          STA A  DACZRO    SEND TO D/A CONV
02130 021E F7 E501          STA B  DACZRO+1  SEND TO D/A CONV
02140 0221 08               INX              INC INDEXED ADDRES
```

```
0222 08               INX
0223 20 ED            BRA    PDFOU2   LOOP AGAIN
[illegible]
                *
0228 27 02            BEQ    PDFOU4   YES. EXIT DISPLAY LOOP
022A 20 CD            BRA    PDFOU1   NO. KEEP SENDING DISPLAY
022C 0F        PDFOU4 SEI
022D CE 4000          LDX    #$4000   RESET INTR GATE.
0230 FF E500          STX    DACZRO
0233 B7 E400          STA A  ADCZRO   RESET ST6800 INT FLIP FLOP
0236 01               NOP
0237 FE 1C9D          LDX    VECSAV   GET ORG INT VEC ADDRESS
023A FF FFF8          STX    IRQVEC   PUT BACK IN INT VEC BUFFER
023D 7E 0100          JMP    DISPL    RET & PRINT COMMAND PROMPT
                *
0240 7F 0248   STOP   CLR    DONTST   CLEAR DONE TEST FLAG
0243 B7 E400          STA A  ADCZRO   CLREA ST6800 IN FLIP FLOP
0246 01               NOP
0247 3B               RTI             RETURN FROM INTERRUPT AND STO
0248 0001      DONTST RMB    1        INTR DONE TEST FLAG
0249 0001      SHFCNT RMB    1        BUF WITH NUM OF SHF FOR 12 BI
024A 0001      MAXCNT RMB    1        TEMP BUF TO COUNT 256 THRU MO
024B 0001      KOUNT  RMB    1        TEMP COUNTER
                *
024C 45        NAMMSG FCC    /ENTER /
0252 04               FCB    4
0253 44        DRVMSG FCC    /DRIVE? /
025A 04               FCB    4
                *
025B 1A07      DISPMS FDB    $1A07,$0C0D,$0A0A,$0A0A,$0A0A,$0A0A
0267 0D0A             FDB    $0D0A
0269 44               FCC    /DATA DISPLAY/
0275 0D0A             FDB    $0D0A,$0D0A
0279 43               FCC    /COMMAND OPTIONS:/
0289 0D0A             FDB    $0D0A,$0D0A
028D 20               FCC    /      0=RETURN TO EKG-EXEC/
02A6 0D0A             FDB    $0D0A,$0D0A
02AA 20               FCC    /      1=PRINT CURRENT MEMORY FILE STA
02D5 0D0A             FDB    $0D0A,$0D0A
02D9 20               FCC    /      2=PRINT PDF AMPLITUDE TABLES/
02FA 0D0A             FDB    $0D0A,$0D0A
02FE 20               FCC    /      3=DISPLAY CH X PDF ON OSCILLISC
0325 0D0A             FDB    $0D0A,$0D0A
0329 20               FCC    /      4=DISPLAY CH Y PDF ON OSCILLISC
0350 0D0A             FDB    $0D0A,$0D0A
0354 20               FCC    /      5=DISPLAY CH Z PDF ON OSCILLISC
037B 0D0A             FDB    $0D0A,$0D0A
037F 20               FCC    /      6=RECONSTRUCT & DISPLAY CH X ON
03A4 4F               FCC    /OSCILLISCOPE/
03B0 0D0A             FDB    $0D0A,$0D0A
03B4 20               FCC    /      7=RECONSTRUCT & DISPLAY CH Y ON
03D9 4F               FCC    /OSCILLISCOPE/
03E5 0D0A             FDB    $0D0A,$0D0A
03E9 20               FCC    /      8=RECONSTRUCT & DISPLAY CH Z ON
```

```
040E 4F                 FCC    /OSCILLISCOPE/
041A 0D0A               FDB    $0D0A,$0D0A
[illegible] 20          FCC    /     [illegible]LOAD MEMORY FILE FROM DISK/
[illegible]             FDB    [illegible]
0443 45                 FCC    /ENTER COMMAND NOW=/
0455 04                 FCB    4
0456 46         FILEMS FCC     /FILE LOADED. /
0463 50         DISMS1 FCC     /PRESS RETURN/
046F 04                 FCB    4
                *
                *
                *FUNCTION       :MAXSCN
                *INPUTS (REG)   :X
                *OUTPUTS        :SHIFT REQUIRED IN SHFCNT
                *CALLS          :NOTHING
                *DESTROYS       :A,B,CC
                *PRUPOSE        :THIS ROUTINE SCANS THE PDF BUFFER PO
                *                TO BY X AND RETURNS THE MINIMUM
                *                SHIFT RIGHTS NECESSARY TO BRING THE
                *                AMPLITUDE COUNTS DOWN TO 11 BITS MAG
                *                FOR OUTPUT FROM THE ST6800 D/A CONVE
                *
                *
0470 DF 19      MAXSCN STX     SAVEX      SAVE PDF ADDR IN X
0472 09                DEX                ADJUST X FOR PROPER INDEXED L
0473 09                DEX
0474 7F 0249           CLR     SHFCNT
0477 7F 024A           CLR     MAXCNT
047A 7C 024A    MAXSC1 INC     MAXCNT     256 BYTES SCANNED YET?
047D 27 1E             BEQ     MAXSC4     YES. EXIT WITH CORRECT SHIFT
047F 08                INX                NO. INC X & LOOK AT NEXT 2 BY
0480 08                INX
0481 A6 00             LDA A   0,X        GET MSB BYTE OF VALUE
0483 81 08             CMP A   #08        IS IT LESS THAN 8?
0485 2D F3             BLT     MAXSC1     YES. NO SHIFT NECESS, GO CHEC
0487 5F                CLR B              NO. SET UP COUNTER
0488 0C                CLC                INSURE CARRY CLEAR
0489 5C         MAXSC2 INC B              NOW SHIFT UNTIL MSB 1 BIT HIT
048A 48                ASL A
048B 25 02             BCS     MAXSC3     CARRY SET?
048D 20 FA             BRA     MAXSC2     NO. KEEP SHIFTING LEFT
048F 17         MAXSC3 TBA                YES. PUT SHIFT COUNT IN A
0490 86 06             LDA A   #6         GET MAX SHIFT
0492 10                SBA                AND GET DIFFERENCE
0493 B1 0249           CMP A   SHFCNT     IS SHIFT LE CURRENT SHIFT COU
0496 2F E2             BLE     MAXSC1     YES. IGNORE THIS VAL[illegible] & CHEC
0498 B7 0249           STA A   SHFCNT     NO. SAVE THIS SHIF[illegible]
049B 20 DD             BRA     MAXSC1     NOW GO CHECK NEXT [illegible]
049D DE 19      MAXSC4 LDX     SAVEX      RETRIEVE INPUT X
049F 39                RTS
                *
                *
                *FUNCTION       :PDFTRN
                *INPUTS         :X
```

```
                   *OUTPUTS         :NONE
                   *CALLS           :NOTHING
                   *DESTROYS        :A,CC
                   *PURPOSE         :THIS ROUTINE TRANSFERS THE PDF DATA
                   *                 POINTED TO BY X INTO A WORK BUFFER
                   *                 AREA AT 3200-33FF.
                   *
                   *
04A0 DF 19         PDFTRN STX    SAVEX      SAVE INDEX IN TEMP BUFFER
04A2 BF 1C96              STS    STKSAV     SAVE STACK POINTER
04A5 35                   TXS               SET UP STACK FOR BUFF TRANSF
04A6 CE 3200              LDX    #$3200     POINT TO WORK BUFF START
04A9 8C 3400       PDFTR1 CPX    #$3400     DONE TRANSFERING YET?
04AC 27 06                BEQ    PDFTR2     YES. RET
04AE 32                   PUL A             NO. GET NEXT BYTE
04AF A7 00                STA A  0,X        STORE IN WORK BUFFER
04B1 08                   INX
04B2 20 F5                BRA    PDFTR1     GO GET NEXT BYTE TO TRANSFER
04B4 DE 19         PDFTR2 LDX    SAVEX      RETRIEVE ENTRY X
04B6 BE 1C96              LDS    STKSAV     RETRIEVE STACK PTR
04B9 39                   RTS
                   *
                   *
                   *FUNCTION        :SCALE
                   *INPUTS          :DATA IN WORK BUFFER
                   *OUTPUTS         :DATA IN WORK BUFFER SCALED BY SHFCNT
                   *CALLS           :NOTHING
                   *DESTROYS        :A,B,CC
                   *PURPOSE         :THIS ROUTINE SCALES THE PDF DATA IN
                   *                 THE WORK BUFFER (3200-33FF) BY THE
                   *                 AMOUNT CALCULATED BY MAXSCN TO ENABL
                   *                 THE ST6800 D/A CONVERTER TO OUTPUT
                   *                 A POSITIVE VOLTAGE PORPORTIONAL TO
                   *                 THE FREQUENCY OF OCCURENCE OF EACH
                   *                 OF THE POSSIBLE 256 AMPLITUDE VALUES
                   *
                   *
04BA DF 19         SCALE  STX    SAVEX      SAVE ENTRY X
04BC CE 3200              LDX    #$3200     LOAD WORK BUFFER START ADDRES
04BF 5F                   CLR B
04C0 B6 0249       SCALE1 LDA A  SHFCNT     PICK UP SCALE SHIFT COUNT
04C3 27 0E                BEQ    SCALE4     IF SHFCNT = 0, NO SCALING NEC
04C5 27 07         SCALE2 BEQ    SCALE3     IS SHFCNT = 0?
04C7 67 00                ASR    0,X        NO. SCALE DATA POINTED TO BY
04C9 66 01                ROR    1,X
04CB 4A                   DEC A
04CC 20 F7                BRA    SCALE2     GO SHIFT DOWN AGAIN
04CE 08            SCALE3 INX               INCREMENT WORK BUFFER DATA PO
04CF 08                   INX
04D0 5C                   INC B             256 2 BYTE WORDS SHIFTED YET?
04D1 26 ED                BNE    SCALE1     NO. KEEP LOADING & SHIFTING
04D3 DE 19         SCALE4 LDX    SAVEX      RETRIEVE ENTRY X
04D5 39                   RTS
                   *
```

```
                 *FUNCTION     :OVRLAY
                 *INPUTS       :X
                 *OUTPUTS      :NONE
                 *CALLS        :ROUTINE POINTED TO BY X
                 *DESTROYS     :ALL REG
                 *PURPOSE      :THIS IS A VECTOR ROUTINE TO CALL OVRLAY
                 * EKG-EXEC TO LOAD ROUTINES FOR EXECUTION.
                 *
04D6 FE 1CAB OVRLAY LDX    OVRLLC
04D9 6E 00          JMP    0,X
                 *
04DB FE 1CA9 PDFPRT LDX    PDFPLC
04DE 6E 00          JMP    0,X
                 *
                 ****************************************************
                 *
                 *        END OF OVERLAY DISPLAY ROUTINES
                 *
                 ****************************************************
                 *
                    END
```

```
             *
             *
             * OVERLAY NAME :PRSTAT
             * [illegible]
             * VERSION : 1:9
             * VERSION DATE : 2 OCT 80
             *
             *
             *
             * PROGRAM DESCRIPTION
             *
             * THIS ROUTINE, WHEN LOADED AND EXECUTED, TAKES
             * THE STATUS DATA CURRENTLY PRESENT IN THE
             * MEMORY FILE HEADER SECTOR AND PRINTS THIS DATA
             * TO THE CONSOLE. THIS ROUTINE CAN BE CALLED
             * IMMEDIATLY FOLLOWING A DATA COLLECTION OR
             * AFTER A PREVIOUSLY COLLECTED DATA FILE HAS BEEN
             * LOADED INTO MEMORY FROM DISK.
             *
             *
             *
             *
             * START OF PRSTAT
             *
             *
             *
0500                ORG    $0500    OVERLAY START ADDRESS
             *
                    OPT    O        ASSB OPT-GEN OBJ FILE
                    OPT    NOG      ASSB OPT-SUPPRESS FCC LIST
             *
             *
             *
             * LABLE DESCRIPTIONS
             *
             * SUPPORT SUBROUTINE ADDRESSES
             *
       CA8F  OUTHCR EQU    $CA8F    EPROMDOS. ALPH STRING TO CONS
       CA36  KEYPDO EQU    $CA36    EPROMDOS. ALPH STRING IN FRM
       1D00  START  EQU    $1D00    EKG-EXEC. SOFT EGK-EXEC START
       2800  DOS    EQU    $2800    DOS ERROR VECTOR, USED FOR JM
             *
             * DATA BUFFERS
             *
       1CA0  LGOFLG EQU    $1CA0    LOAD VERSUS EXECUTE FLAG
       1CA5  HXASLC EQU    $1CA5    HXASC ADDR PASS BUFFER
       1CA7  HXBUF  EQU    $1CA7    HXASC PARAMETER BUFFER
       1C96  STKSAV EQU    $1C96    TEMP STACK SAVE BUFFER
       3400  HDRSTR EQU    $3400    MEM FILE HEADER START
       3490  LOOPCT EQU    $3490    TOTL CAL LOOPS EXEC IN WAIT
       3494  SAMPNO EQU    $3494    NUM OF SAMPLES TAKEN
       3496  LPCAL  EQU    $3496    NUM OF CAL LOOPS PER 1 INTR
       3497  MAXZ   EQU    $3497    MAX VLU IN CH Z
       3498  MAXZLO EQU    $3498    LOCATION OF MAX VLU IN CH Z
```

```
          349A     MINZ   EQU   $349A     MIN VLU IN CH Z
          349B     MINZLO EQU   $349B     LOCATION OF MIN VLU IN CH Z
          [illegible]                     MAX VLU IN CH Y
          [illegible]                     [illegible] VLU [illegible] CH Y
          34A0     MINY   EQU   $34A0     MIN VLU IN CH Y
          34A1     MINYLO EQU   $34A1     LOCATION OF MIN VLU IN CH Y
          34A3     MAXX   EQU   $34A3     MAX VLU IN CH X
          34A4     MAXXLO EQU   $34A4     LOCATION OF MAX VLU IN CH X
          34A6     MINX   EQU   $34A6     MIN VLU IN CH X
          34A7     MINXLO EQU   $34A7     LOCATION OF MIN VLU IN CH X
          34A9     MEMBIT EQU   $34A9     NUM OF BITS AVAILABLE FOR STO
          34AC     DTABIT EQU   $34AC     NUM OF BITS USED TO STOR DTA
          34B0     XBITS  EQU   $34B0     NUM OF BITS USED TO STOR X DT
          34B3     YBITS  EQU   $34B3     NUM OF BITS USED TO STOR Y DT
          34B6     ZBITS  EQU   $34B6     NUM OF BITS USED TO STOR Z DT
          34B9     TBITS  EQU   $34B9     NUM OF BITS USED TO STOR TIME
          34BC     ACELCT EQU   $34BC     # BITS FED TO VAR LEN CODER
          34C2     BASSAV EQU   $34C2     SAVE MEM LOC $3620 FROM BASIC
          34C3     ENTRPY EQU   $34C3     START OF ENTROPY BUFFER
          3460     MAXMIN EQU   $3460     START OF MAX,MIN ASCII DATA
                   *
                   *
                   *FUNCTION :PRSTAT
                   *INPUTS (REG) :NONE
                   *OUTPUTS (REG):NONE
                   *CALLS :OUTPUT,HXASC
                   *DESTROYS(REG):A,B,CC,X
                   *PURPOSE : TO LIST STATUS INFORMATION FROM DATA
                   * COLLECTION FILE TO TERMINAL.
                   *
                   *
0500 86 08         PRSTAT LDA A #8        CLR MSG FIELDS IN STAT OUTPUT
0502 CE 0A66              LDX   #NAME
0505 BD 0930              JSR   MSGCLR
0508 86 35                LDA A #53
050A CE 0A8C              LDX   #SUBJ
050D BD 0930              JSR   MSGCLR
0510 86 0A                LDA A #10
0512 CE 0ADF              LDX   #RATE
0515 BD 0930              JSR   MSGCLR
0518 86 14                LDA A #20
051A CE 0B07              LDX   #DATE
051D BD 0930              JSR   MSGCLR
0520 86 08                LDA A #8
0522 CE 0B39              LDX   #TIME
0525 BD 0930              JSR   MSGCLR
0528 86 19                LDA A #25
052A CE 0B5F              LDX   #TYPE
052D BD 0930              JSR   MSGCLR
0530 86 35                LDA A #53
0532 CE 0F29              LDX   #COMMNT
0535 BD 0930              JSR   MSGCLR
0538 86 08                LDA A #8
053A CE 0B96              LDX   #XENT
```

```
053D BD 0930        JSR    MSGCLR
[illegible] 86 08   LDA A  #8
[illegible]
0545 BD 0930        JSR    MSGCLR
0548 86 08          LDA A  #8
054A CE 0BFA        LDX    #ZFNT
054D BD 0930        JSR    MSGCLR
0550 86 08          LDA A  #8
0552 CE 0C2C        LDX    #TFNT
0555 BD 0930        JSR    MSGCLR
0558 86 08          LDA A  #8
055A CE 0CCE        LDX    #MANCPR
055D BD 0930        JSR    MSGCLR
0560 86 08          LDA A  #8
0562 CE 0D11        LDX    #CPTACH
0565 BD 0930        JSR    MSGCLR
0568 86 05          LDA A  #5
056A CE 0D57        LDX    #CPREF
056D BD 0930        JSR    MSGCLR
0570 86 05          LDA A  #5
0572 CE 0D99        LDX    #TINTF
0575 BD 0930        JSR    MSGCLR
0578 86 05          LDA A  #5
057A CE 0DCB        LDX    #COLDGR
057D BD 0930        JSR    MSGCLR
0580 86 08          LDA A  #8
0582 CE 0E2F        LDX    #AMINX
0585 BD 0930        JSR    MSGCLR
0588 86 08          LDA A  #8
058A CE 0DFD        LDX    #AMAXX
058D BD 0930        JSR    MSGCLR
0590 86 08          LDA A  #8
0592 CE 0E93        LDX    #AMINY
0595 BD 0930        JSR    MSGCLR
0598 86 08          LDA A  #8
059A CE 0E61        LDX    #AMAXY
059D BD 0930        JSR    MSGCLR
05A0 86 08          LDA A  #8
05A2 CE 0E93        LDX    #AMINY
05A5 BD 0930        JSR    MSGCLR
05A8 86 08          LDA A  #8
05AA CE 0EC5        LDX    #AMAXZ
05AD BD 0930        JSR    MSGCLR
05B0 86 08          LDA A  #8
05B2 CE 0EF7        LDX    #AMINZ
05B5 BD 0930        JSR    MSGCLR
05B8 CE 3402        LDX    #HDRBFR+2   NOW PUT DPA IN FIL HDR IN M
05BB FF 095D        STX    FRMLOC
05BE CE 0A66        LDX    #NAME
05C1 BD 093C        JSR    MSGPUT
05C4 CE 0A8C        LDX    #SUBJ
05C7 BD 093C        JSR    MSGPUT
05CA CE 0A9C        LDX    #DATE
05CD BD 093C        JSR    MSGPUT
```

```
05D0 CE 0F07          LDX   #DATE
05D3 BD 093C          JSR   MSGPUT
[illegible] CE 0F39   LDX   #T[illegible]
[illegible]
05DC CE 0F29          LDX   #COMENT
05DF BD 093C          JSR   MSGPUT
05E2 FE 34C3          LDX   ENTRPY    GET LOC OF ENTROPY BUFFER
05E5 8C 2020          CPX   #$2020    CHK IF ENTROPY CALC MADE?
05E8 27 03            BEQ   PRST00    NO. STORE NOT CALC MSG
05EA 7E 06A4          JMP   PRSTA0    YES. GET ASCII STRING FROM HD
05ED CE 09FD PRST00   LDX   #ENTCLR
05F0 FF 095D          STX   FRMLOC
05F3 CE 0B96          LDX   #XENT
05F6 BD 093C          JSR   MSGPUT
05F9 CE 09FD          LDX   #ENTCLR
05FC FF 095D          STX   FRMLOC
05FF CE 0BC8          LDX   #YENT
0602 BD 093C          JSR   MSGPUT
0605 CE 09FD          LDX   #ENTCLR
0608 FF 095D          STX   FRMLOC
060B CE 0BFA          LDX   #ZENT
060E BD 093C          JSR   MSGPUT
0611 CE 09FD          LDX   #ENTCLR
0614 FF 095D          STX   FRMLOC
0617 CE 0C2C          LDX   #TENT
061A BD 093C          JSR   MSGPUT
061D CE 09FD          LDX   #ENTCLR
0620 FF 095D          STX   FRMLOC
0623 CE 0CCE          LDX   #MAXCPR
0626 BD 093C          JSR   MSGPUT
0629 CE 09FD          LDX   #ENTCLR
062C FF 095D          STX   FRMLOC
062F CE 0D11          LDX   #CPRACH
0632 BD 093C          JSR   MSGPUT
0635 CE 09FD          LDX   #ENTCLR
0638 FF 095D          STX   FRMLOC
063B CE 0D57          LDX   #CPREFF
063E BD 093C          JSR   MSGPUT
0641 CE 09FD          LDX   #ENTCLR
0644 FF 095D          STX   FRMLOC
0647 CE 0D99          LDX   #TIMEFF
064A BD 093C          JSR   MSGPUT
064D CE 09FD          LDX   #ENTCLR
0650 FF 095D          STX   FRMLOC
0653 CE 0DCB          LDX   #COLDUR
0656 BD 093C          JSR   MSGPUT
0659 CE 09FD          LDX   #ENTCLR
065C FF 095D          STX   FRMLOC
065F CE 0DFD          LDX   #AMAXX
0662 BD 093C          JSR   MSGPUT
0665 CE 09FD          LDX   #ENTCLR
0668 FF 095D          STX   FRMLOC
066B CE 0E2F          LDX   #AMINX
066E BD 093C          JSR   MSGPUT
```

```
02190 0671 CE 09FD          LDX    #ENTCLR
02200 0674 FF 095D          STX    FRMLOC
02210 [illegible]
02220 067A BD 093C          JSR    [illegible]
02230 067D CE 09FD          LDX    #ENTCLR
02240 0680 FF 095D          STX    FRMLOC
02250 0683 CE 0E93          LDX    #AMINY
02260 0686 BD 093C          JSR    MSGPUT
02270 0689 CE 09FD          LDX    #ENTCLR
02280 068C FF 095D          STX    FRMLOC
02290 068F CE 0EC5          LDX    #AMAXZ
02300 0692 BD 093C          JSR    MSGPUT
02310 0695 CE 09FD          LDX    #ENTCLR
02320 0698 FF 095D          STX    FRMLOC
02330 069B CE 0EF7          LDX    #AMINZ
02340 069E BD 093C          JSR    MSGPUT
02350 06A1 7E 070A          JMP    PRSTA1
02360 06A4 CE 34C3 PRSTA0   LDX    #ENTRPY   GET ADDRESS OF ENTRPY BUFFER
02370 06A7 FF 095D          STX    FRMLOC
02380 06AA CE 0B96          LDX    #XENT     NOW PRINT APPROPRIATE DAT TO
02390 06AD BD 093C          JSR    MSGPUT
02400 06B0 CE 0BC8          LDX    #YENT
02410 06B3 BD 093C          JSR    MSGPUT
02420 06B6 CE 0BFA          LDX    #ZENT
02430 06B9 BD 093C          JSR    MSGPUT
02440 06BC CE 0C2C          LDX    #TENT
02450 06BF BD 093C          JSR    MSGPUT
02460 06C2 CE 0CCE          LDX    #MAXCPR
02470 06C5 BD 093C          JSR    MSGPUT
02480 06C8 CE 0D11          LDX    #CPRACH
02490 06CB BD 093C          JSR    MSGPUT
02500 06CE CE 0D57          LDX    #CPRTTF
02510 06D1 BD 093C          JSR    MSGPUT
02520 06D4 CE 0D99          LDX    #TIMEFF
02530 06D7 BD 093C          JSR    MSGPUT
02540 06DA CE 0DCB          LDX    #COLDUR
02550 06DD BD 093C          JSR    MSGPUT
02560 06E0 CE 3460          LDX    #MAXMIN   NOW GET ADDR WHERE ASCII MAX,
02570 06E3 FF 095D          STX    FRMLOC
02580 06E6 CE 0DFD          LDX    #AMAXX
02590 06E9 BD 093C          JSR    MSGPUT
02600 06EC CE 0E2F          LDX    #AMINX
02610 06EF BD 093C          JSR    MSGPUT
02620 06F2 CE 0E61          LDX    #AMAXY
02630 06F5 BD 093C          JSR    MSGPUT
02640 06F8 CE 0E93          LDX    #AMINY
02650 06FB BD 093C          JSR    MSGPUT
02660 06FE CE 0EC5          LDX    #AMAXZ
02670 0701 BD 093C          JSR    MSGPUT
02680 0704 CE 0EF7          LDX    #AMINZ
02690 0707 BD 093C          JSR    MSGPUT
02700 070A FE 3400 PRSTA1   LDX    HDRSTR    NOW DECODE COMPRS TYPE & PUT
02710 070D 8C 4E43          CPX    #$4E43    (NC) NOCPRS?
02720 0710 26 08            BNE    PRSTA2    NO
```

```
[illegible] CE 09?9          LDX    #MOCCMP   YES. POINT TO [illegible] MSG
0715 FF 095D                 STX    FRMLOC
[illegible] 20 42            BRA    PRSTA7
[illegible] 8C 5441 PRSTA2   CPX    #$5441    (TA) TOLAN-A?
071D 26 08                   BNE    PRSTA3    NO.
071F CE 09C2                 LDX    #TOLA     YES. POINT TO TOLAN-A MSG
0722 FF 095D                 STX    FRMLOC
0725 20 35                   BRA    PRSTA7
0727 8C 5442 PRSTA3          CPX    #$5442    (TB) TOLAN-B?
072A 26 08                   BNE    PRSTA4    NO
072C CE 09CA                 LDX    #TOLB     YES. POINT TO TOLAN-B MSG
072F FF 095D                 STX    FRMLOC
0732 20 28                   BRA    PRSTA7
0734 8C 444F PRSTA4          CPX    #$444F    (DO) DOVER?
0737 26 08                   BNE    PRSTA5    NO
0739 CE 09D2                 LDX    #DOVR     YES. POINT TO DOVER MSG
073C FF 095D                 STX    FRMLOC
073F 20 1B                   BRA    PRSTA7
0741 8C 5450 PRSTA5          CPX    #$5450    (TP) TURNING POINT?
0744 26 08                   BNE    PRSTA6    NO.
0746 CE 09D8                 LDX    #TURNPT   YES. POINT TO TURNING POINT M
0749 FF 095D                 STX    FRMLOC
074C 20 0E                   BRA    PRSTA7
074E 8C 494E PRSTA6          CPX    #$494E    (IN) 2ND ORDER INTERPOLATOR?
0751 27 03                   BEQ    PRST61    YES. TYPE IS RECOGNIZED. CONT
0753 7E 095F                 JMP    ERROR     NO. TYPE NOT RECOGNIZED. ERR
0756 CE 09E6 PRST61          LDX    #INTER
0759 FF 095D                 STX    FRMLOC
075C CE 0B5F PRSTA7          LDX    #TYPE     PUT CONFRS TYPE IN MSG
075F BD 093C                 JSR    MSGPUT
0762 CE 0A12                 LDX    #HDRMSG   NOW PT TO STATMSG PAGE 1
0765 BD CA8F                 JSR    OUTNCR    AND PRINT PAGE 1
0768 BD CA36                 JSR    KEYBD0
076B CE 0C6B                 LDX    #HDRMS2   NOW PT TO STATMSG PAGE 2
076E BD CA8F                 JSR    OUTNCR    AND PRINT PAGE 2
0771 BD CA36                 JSR    KEYBD0
0774 CE 0FD4                 LDX    #NUMSMP+2  PICK UP NUM OF SMPLS,
0777 B6 3495                 LDA A  SAMPNO+1  CONVRT TO ASCII, & STORE
077A BD 0955                 JSR    HXASC     TO OUTPUT STRING.
077D CE 0FD2                 LDX    #NUMSMP
0780 B6 3494                 LDA A  SAMPNO
0783 BD 0955                 JSR    HXASC
0786 CE 1019                 LDX    #NUMLPS   PICK UP VLU OF CAL TST
0789 B6 3496                 LDA A  LPCAL     & STORE TO OUTPUT STRING
078C BD 0955                 JSR    HXASC
078F CE 1060                 LDX    #TOTLUP+6  PICK UP TOTL LOOP COUNT
0792 B6 3493                 LDA A  LOOPCT+3  ,CONVERT TO ASCII & STORE
0795 BD 0955                 JSR    HXASC     TO OUTPUT STRING
0798 CE 105E                 LDX    #TOTLUP+4
079B B6 3492                 LDA A  LOOPCT+2
079E BD 0955                 JSR    HXASC
07A1 CE 105C                 LDX    #TOTLUP+2
07A4 B6 3491                 LDA A  LOOPCT+1
07A7 BD 0955                 JSR    HXASC
```

```
07AA CE 105A      LDX    #TOTLUP
07AD B6 3490      LDA A  LOOPCT
[illegible]       JSR    HXASC
07B3 CE 10C5      LDX    #XMAXNM   PICK UP MAX VLU ON CH X
07B6 B6 34A3      LDA A  MAXX      CON TO ASCII & STR IN
07B9 BD 0955      JSR    HXASC     TO OUTPUT STRING
07BC CE 10E1      LDX    #XMAXLO+2
07BF B6 34A5      LDA A  MAXXLO+1
07C2 BD 0955      JSR    HXASC
07C5 CE 10DF      LDX    #XMAXLO
07C8 B6 34A4      LDA A  MAXXLO
07CB BD 0955      JSR    HXASC
07CE CE 10F2      LDX    #XMINNM   PICK UP MIN VLU ON CH X
07D1 B6 34A6      LDA A  MINX      CONV TO ASCII & STORE IN
07D4 BD 0955      JSR    HXASC     OUTPUT STRING
07D7 CE 110E      LDX    #XMINLO+2
07DA B6 34A8      LDA A  MINXLO+1
07DD BD 0955      JSR    HXASC
07E0 CE 110C      LDX    #XMINLO
07E3 B6 34A7      LDA A  MINXLO
07E6 BD 0955      JSR    HXASC
07E9 CE 111F      LDX    #YMAXNM   PICK UP MAX VLU ON CH Y
07EC B6 349D      LDA A  MAXY      CONV TO ASCII & STORE IN
07EF BD 0955      JSR    HXASC     OUTPUT STRING
07F2 CE 113B      LDX    #YMAXLO+2
07F5 B6 349F      LDA A  MAXYLO+1
07F8 BD 0955      JSR    HXASC
07FB CE 1139      LDX    #YMAXLO
07FE B6 349E      LDA A  MAXYLO
0801 BD 0955      JSR    HXASC
0804 CE 114C      LDX    #YMINNM   PICK UP MIN VLU ON CH Y
0807 B6 34A0      LDA A  MINY      CONV TO ASCII & STORE IN
080A BD 0955      JSR    HXASC     OUTPUT STRING
080D CE 1168      LDX    #YMINLO+2
0810 B6 34A2      LDA A  MINYLO+1
0813 BD 0955      JSR    HXASC
0816 CE 1166      LDX    #YMINLO
0819 B6 34A1      LDA A  MINYLO
081C BD 0955      JSR    HXASC
081F CE 1179      LDX    #ZMAXNM   PICK UP MAX VLU ON CH Z
0822 B6 3497      LDA A  MAXZ      CONV TO ASCII & STORE IN
0825 BD 0955      JSR    HXASC     OUTPUT STRING
0828 CE 1195      LDX    #ZMAXLO+2
082B B6 3499      LDA A  MAXZLO+1
082E BD 0955      JSR    HXASC
0831 CE 1193      LDX    #ZMAXLO
0834 B6 3498      LDA A  MAXZLO
0837 BD 0955      JSR    HXASC
083A CE 11A6      LDX    #ZMINNM   PICK UP MIN VLU ON CH Z
083D B6 349A      LDA A  MINZ      CONV TO ASCII $ STORE IN
0840 BD 0955      JSR    HXASC     OUPUT STRING
0843 CE 11C2      LDX    #ZMINLO+2
0846 B6 349C      LDA A  MINZLO+1
0849 BD 0955      JSR    HXASC
```

```
084C CE 11C0      LDX     #ZMINLO
084F B6 349B      LDA A   MINZLO
[illegible] 0955  JSR     [illegible]
[illegible] CE 120E  LDX  [illegible]
0858 B6 34AB      LDA A   MEMBIT+2 AVAILABLE FOR DATA STOR
085B BD 0955      JSR     HXASC     , CONV TO ASCII & STOR
085E CE 120C      LDX     #AVABIT+2  IN OUTPUT STRING
0861 B6 34AA      LDA A   MEMBIT+1
0864 BD 0955      JSR     HXASC
0867 CE 120A      LDX     #AVABIT
086A B6 34A9      LDA A   MEMBIT
086D BD 0955      JSR     HXASC
0870 CE 124B      LDX     #TCPBIT+6
0873 B6 34BF      LDA A   ACELCT+3
0876 BD 0955      JSR     HXASC
0879 CE 1249      LDX     #TCPBIT+4
087C B6 34BE      LDA A   ACELCT+2
087F BD 0955      JSR     HXASC
0882 CE 1247      LDX     #TCPBIT+2
0885 B6 34BD      LDA A   ACELCT+1
0888 BD 0955      JSR     HXASC
088B CE 1245      LDX     #TCPBIT
088E B6 34BC      LDA A   ACELCT
0891 BD 0955      JSR     HXASC
0894 CE 127F      LDX     #TOTBIT+6  PICK UP NUM OF BITS
0897 B6 34AF      LDA A   DTABIT+3 ACTUALL STORED
089A BD 0955      JSR     HXASC    BECAUSE OF DATA
089D CE 127D      LDX     #TOTBIT+4  COMPRESSION, CONV TO
08A0 B6 34AE      LDA A   DTABIT+2 ASCII & STORE IN
08A3 BD 0955      JSR     HXASC    OUTPUT STRING
08A6 CE 127B      LDX     #TOTBIT+2
08A9 B6 34AD      LDA A   DTABIT+1
08AC BD 0955      JSR     HXASC
08AF CE 1279      LDX     #TOTBIT
08B2 B6 34AC      LDA A   DTABIT
08B5 BD 0955      JSR     HXASC
08B8 CE 12B7      LDX     #XBITNM+4  PICK UP NUM OF BITS
08BB B6 34B2      LDA A   XBITS+2  ACTUALLY STORED FOR
08BE BD 0955      JSR     HXASC    DATA IN CHANNEL X,
08C1 CE 12B5      LDX     #XBITNM+2  CONV TO ASCII &
08C4 B6 34B1      LDA A   XBITS+1  STORE TO OUTPUT STRING
08C7 BD 0955      JSR     HXASC
08CA CE 12B3      LDX     #XBITNM
08CD B6 34B0      LDA A   XBITS
08D0 BD 0955      JSR     HXASC
08D3 CE 12BF      LDX     #YBITNM+4  PICK UP NUM OF BITS
08D6 B6 34B5      LDA A   YBITS+2  ACTUALLY STORED FOR
08D9 BD 0955      JSR     HXASC    DATA IN CHANNEL Y,
08DC CE 12BD      LDX     #YBITNM+2  CONV TO ASCII &
08DF B6 34B4      LDA A   YBITS+1  STORE IN OUTPUT STRING
08E2 BD 0955      JSR     HXASC
08E5 CE 12BB      LDX     #YBITNM
08E8 B6 34B3      LDA A   YBITS
08EB BD 0955      JSR     HXASC
```

```
08EE CE 1327        LDX     #ZBITNM+4  PICK UP NUM OF BITS
08F1 B6 34B8        LDA A   ZBITS+2  ACTUALLY STORED FOR
08F4 BD 0955        JSR     HXASC     DATA IN CH[illegible],
08F7 CE 1325        LDX     #ZBITNM+2  CONV TO ASCII &
08FA B6 34B7        LDA A   ZBITS+1  STORE IN OUTPUT STRING
08FD BD 0955        JSR     HXASC
0900 CE 1323        LDX     #ZBITNM
0903 B6 34B6        LDA A   ZBITS
0906 BD 0955        JSR     HXASC
0909 CE 136D        LDX     #TBITNM+4  PICK UP NUM OF BITS
090C B6 34BB        LDA A   TBITS+2  ACTUALLY STORED FOR
090F BD 0955        JSR     HXASC     TIME (OR OTHER PARAMETER)
0912 CE 136B        LDX     #TBITNM+2  CONV TO ASCII &
0915 B6 34BA        LDA A   TBITS+1  STORE IN OUTPUT STRING
0918 BD 0955        JSR     HXASC
091B CE 1369        LDX     #TBITNM
091E B6 34B9        LDA A   TBITS
0921 BD 0955        JSR     HXASC
0924 CE 0F89        LDX     #COSTAT  GET OUTPUT STRING ADDR
0927 BD CA8F        JSR     OUTNCR   & SEND TO CONSOLE
092A BD CA36        JSR     KEYBD0   WAIT FOR TERM INPUT
092D 7E 2800        JMP     DOS      JUMP BACK TO CALLING ROUTINE
                *
                *FUNC: MSGCLR
                *INPUTS: A (# OF SPACES TO CLR),X (LOC TO PUT SPACES
                *OUTPUTS: ASCII $20 TO MEM AT X
                *CALLS: NOTHING
                *DESTROYS: A,B,CC
                *PURPOSE: THIS ROUTINE CLRS THE MSG BUFFER EACH TIME
                * PRSTAT IS CALLED FOR NEW INFO OFF OF THE MEM DATA
                *
0930 4C         MSGCLR INC A            INC COUNTER
0931 C6 20             LDA B  #$20      ASCII SPACE
0933 4A         MSGCL1 DEC A
0934 27 05             BEQ    MSGCL2    IS BUFFER CLRED?
0936 E7 00             STA B  0,X       NO. KEEP LOOPING
0938 08                INX
0939 20 F8             BRA    MSGCL1
093B 39         MSGCL2 RTS
                *
                * END OF MSGCLR
                *
                *******************
                *
                *FUNC: MSGPUT
                *INPUTS: X(ADDR WHERE DATA IN FRMLOC GOING)
                *OUTPUTS: ASCII DATA TO ADDR IN X
                *CALLS: NOTHING
                *DESTROYS: A,X,CC
                *PURPOSE: THIS ROUTINES TRANSFER ASCII TEXT
                * FROM MEM FILE HDR TO STAT MSG BUFFER
                *
093C BF 1C96    MSGPUT STS    STKSAV    SAVE CURRENT STACK PTR
093F BE 095D           LDS    FRMLOC    GET LOC OF DATA FOR [illegible]
```

```
0942 34                  DES
0943 32          MSGPU1  PUL A
[illegible]4 81 04       CMP A   #4        END OF MSG?
[illegible]6 27 05       BEQ     MSGPU2    YES. EXIT
0948 A7 00               STA A   0,X       NO. PUT CHAR IN MSG
094A 08                  INX
094B 20 F6               BRA     MSGPU1
094D 31          MSGPU2  INS               UPDATE STK WITH NEXT TRNSFR L
094E BF 095D             STS     FRMLOC    STR BACK IN FRMLOC
0951 BE 1C96             LDS     STKSAV    PICK UP ENTRY STACK PTR
0954 39                  RTS
                 *
                 * END MSGPUT
                 *
                 ************
                 *
                 *FUNC: HXASC
                 *INPUTS:A,X (DATA TO BE CONV'D,ADDR TO STORE ASCII O
                 *CALLS: HSASC ROUTINE IN EKG-EXEC
                 *DISTROYS: A,X,CC
                 *PURPOSE: THIS IS A RELOC PASS ROUTINE TO PICK UP
                 * ADDRO OF HXASC IN EKG-EXEC AND JUMP TO IT
                 *
0955 FF 1CA7     HXASC   STX     HXBUF     SAVE X IN PARAMATER BUFFER
0958 FE 1CA5             LDX     HXASLC    GET ADDR OF CONV ROUTINE IN E
095B 6E 00               JMP     0,X       JUMP TO IT
                 *
                 * END OF HXASC
                 *
                 **************
                 *
095D 0002        FRMLOC  RMB     2         DATA FROM LOC FOR MSGPUT
                 *
095F CE 096B     ERROR   LDX     #ERRMSG
0962 BD CA8F             JSR     OUTNCR
0965 BD CA36             JSR     KEYEDO
0968 7E 1D00             JMP     START
                 *
096B 0707        ERRMSG  FDB     $0707,$0D0A
096F 4D                  FCC     /MEMORY FILE COMPRESSION TYPE /
098C 4E                  FCC     /NOT RECOGNIZED. PRESS RETURN/
09A8 04                  FCB     4
                 *
                 *************************************************
                 * END PRSTAT
                 *************************************************
                 *
                 *************************************************
                 *
                 * OUTPUT STRING TO LIST TO CONSOLE
                 *
                 *************************************************
                 *
09A9 4E          NOCOMP  FCC     /NO COMPRESSION PERFORMED/
```

```
09C1 04              FCB    4
09C2 54       TOLA   FCC    /TOLAN-A/
09C9 04              FCB    4
09CA 54       TOLB   FCC    /TOLAN-B/
09D1 04              FCB    4
09D2 44       DOWR   FCC    /DOWER/
09D7 04              FCB    4
09D8 54       TURNPT FCC    /TURNING POINT/
09E5 04              FCB    4
09E6 32       INTER  FCC    /2ND ORDER INTERPOLATOR/
09FC 04              FCB    4
09FD 43       ENTCLR FCC    /CALCULATION NOT MADE/
0A11 04              FCB    4
0A12 1A07     HDRMSG FDB    $1A07,$0C0D,$0A0A,$0A0A,$0A0A,$0A0A
0A1E 3A              FCC    /:EKG SAMPLE COLLECTION STATISTICS/
0A3F 20              FCC    / : PAGE 1/
0A48 0D0A            FDB    $0D0A,$0D0A
0A4C 46              FCC    /FILENAME. . . . . . . . . . /
0A66 0008     NAME   RMB    8
0A6E 0D0A            FDB    $0D0A,$0D0A
0A72 53              FCC    /SUBJECT . . . . . . . . . . /
0A8C 0035     SUBJ   RMB    53
0AC1 0D0A            FDB    $0D0A,$0D0A
0AC5 53              FCC    /SAMPLING RATE . . . . . . . /
0ADF 000A     RATE   RMB    10
0AE9 0D0A            FDB    $0D0A,$0D0A
0AED 44              FCC    /DATE OF COLLECTION. . . . /
0B07 0014     DATE   RMB    20
0B1B 0D0A            FDB    $0D0A,$0D0A
0B1F 54              FCC    /TIME OF COLLECTION. . . . /
0B39 0008     TIME   RMB    8
0B41 0D0A            FDB    $0D0A,$0D0A
0B45 43              FCC    /COMPRESSION USED. . . . . /
0B5F 0019     TYPE   RMB    25
0B78 0D0A            FDB    $0D0A,$0D0A
0B7C 43              FCC    /CHANNEL X ENTROPY . . . . /
0B96 0008     XENT   RMB    8
0B9E 20              FCC    / BITS         /
0BAA 0D0A            FDB    $0D0A,$0D0A
0BAE 43              FCC    /CHANNEL Y ENTROPY . . . . /
0BC8 0008     YENT   RMB    8
0BD0 20              FCC    / BITS         /
0BDC 0D0A            FDB    $0D0A,$0D0A
0BE0 43              FCC    /CHANNEL Z ENTROPY . . . . /
0BFA 0008     ZENT   RMB    8
0C02 20              FCC    / BITS         /
0C0E 0D0A            FDB    $0D0A,$0D0A
0C12 54              FCC    /TOTAL SOURCE ENTROPY. . . /
0C2C 0008     TENT   RMB    8
0C34 20              FCC    / BITS         /
0C40 0D0A            FDB    $0D0A,$0D0A
0C44 50              FCC    /PRESS RETURN FOR PAGE 2 OF STATISTIC
0C6A 04              FCB    4
0C6B 1A07     HDRMS2 FDB    $1A07,$0D0A
```

```
0C6F 45                FCC    /EKG SAMPLE COLLECTION STATISTICS/
0C8F 20                FCC    / : PAGE 2/
[illegible]            FDB    [illegible]
0C9C 41                FCC    /[illegible]/
0CB2 0D0A              FDB    $0D0A
0CB4 52                FCC    /RATIO POSSIBLE. . . . . . /
0CCE 0008      MAXCPR  RMB    8
0CD6 20                FCC    / : 1          /
0CE2 0D0A              FDB    $0D0A
0CE4 43                FCC    /COMPRESSION RATIO/
0CF5 0D0A              FDB    $0D0A
0CF7 41                FCC    /ACHIEVED. . . . . . . . . . /
0D11 0008      CPRACH  RMB    8
0D19 20                FCC    / : 1          /
0D25 0D0A              FDB    $0D0A
0D27 41                FCC    /ACHIEVED COMPRESSION/
0D3B 0D0A              FDB    $0D0A
0D3D 45                FCC    /EFFICIENCY. . . . . . . . . /
0D57 0005      CPREFF  RMB    5
0D5C 20                FCC    / % OF MAXIMUM  /
0D6B 0D0A              FDB    $0D0A
0D6D 43                FCC    /COMPRESSION TIME/
0D7D 0D0A              FDB    $0D0A
0D7F 45                FCC    /EFFICIENCY OBTAINED . . . /
0D99 0005      TIMEFF  RMB    5
0D9E 20                FCC    / % SMP INTERVAL/
0DAD 0D0A              FDB    $0D0A,$0D0A
0DB1 43                FCC    /COLLECTION DURATION . . . /
0DCB 0005      COLDUR  RMB    5
0DD0 20                FCC    / SECONDS       /
0DDF 0D0A              FDB    $0D0A,$0D0A
0DE3 43                FCC    /CHANNEL X MAXIMUM . . . . /
0DFD 0008      AMAXX   RMB    8
0E05 20                FCC    / VOLTS         /
0E11 0D0A              FDB    $0D0A,$0D0A
0E15 43                FCC    /CHANNEL X MINIMUM . . . . /
0E2F 0008      AMINX   RMB    8
0E37 20                FCC    / VOLTS         /
0E43 0D0A              FDB    $0D0A,$0D0A
0E47 43                FCC    /CHANNEL Y MAXIMUM . . . . /
0E61 0008      AMAXY   RMB    8
0E69 20                FCC    / VOLTS         /
0E75 0D0A              FDB    $0D0A,$0D0A
0E79 43                FCC    /CHANNEL Y MINIMUM . . . . /
0E93 0008      AMINY   RMB    8
0E9B 20                FCC    / VOLTS         /
0EA7 0D0A              FDB    $0D0A,$0D0A
0EAB 43                FCC    /CHANNEL Z MAXIMUM . . . . /
0EC5 0008      AMAXZ   RMB    8
0ECD 20                FCC    / VOLTS         /
0ED9 0D0A              FDB    $0D0A,$0D0A
0EDD 43                FCC    /CHANNEL Z MINIMUM . . . . /
0EF7 0008      AMINZ   RMB    8
0EFF 20                FCC    / VOLTS         /
```

```
0F0B CD0A          FDB    $0D0A,$0D0A
0F0F 43            FCC    /COMMENTS. . . . . . . . . /
[illegible]
[illegible]
0F62 50            FCC    /PRESS RETURN FOR PAGE 3 OF STATISTIC
0F88 04            FCB    4
0F89 1A07   COSTAT FDB    $1A07,$0C0D,$0A0A,$0A0A,$0A0A,$0A0A
0F95 45            FCC    /EKG SAMPLE COLLECTION STATISTICS: PA
0FBD 0D0A          FDB    $0D0A    CR/LF
0FBF 20            FCC    / NUMBER OF SAMPLES TAKEN /
0FD8 28            FCC    /(SAMPNO) = /
0FE2 0004   NUMSMP RMB    4
0FE6 20            FCC    / (HEX)/
0FEC 0D0A          FDB    $0D0A    CR/LF
0FEE 20            FCC    / MAXIMUM LOOP COUNT PER/
1005 20            FCC    / INTERRUPT (LPCAL) = /
1019 0002   NUMLPS RMB    2
101B 20            FCC    / (HEX)/
1021 0D0A          FDB    $0D0A    CR/LF
1023 20            FCC    / TOTAL WAITING LOOP COUNTS DURING/
1044 20            FCC    / COLLECTION (LOOPCT) = /
105A 0008   TOTLUP RMB    8
1062 20            FCC    / (HEX)/
1068 0D0A          FDB    $0D0A
106A 54            FCC    /TIME EFFICIENCY = (1-(LOOPCT/(SAMPNO
1097 2A            FCC    /*100/
109B 0D0A          FDB    $0D0A,$0D0A
109F 43            FCC    /CHANNEL MAXIMUMS AND MINIMUMS/
10BC 0D0A          FDB    $0D0A
10BE 20            FCC    / XMAX= /
10C5 0002   XMAXNM RMB    2
10C7 20            FCC    / (HEX) AT SAMPLE NUMBER /
10DF 0004   XMAXLO RMB    4
10E3 20            FCC    / (HEX)/
10E9 0D0A          FDB    $0D0A
10EB 20            FCC    / XMIN= /
10F2 0002   XMINNM RMB    2
10F4 20            FCC    / (HEX) AT SAMPLE NUMBER /
110C 0004   XMINLO RMB    4
1110 20            FCC    / (HEX)/
1116 0D0A          FDB    $0D0A
1118 20            FCC    / YMAX= /
111F 0002   YMAXNM RMB    2
1121 20            FCC    / (HEX) AT SAMPLE NUMBER /
1139 0004   YMAXLO RMB    4
113D 20            FCC    / (HEX)/
1143 0D0A          FDB    $0D0A
1145 20            FCC    / YMIN= /
114C 0002   YMINNM RMB    2
114E 20            FCC    / (HEX) AT SAMPLE NUMBER /
1166 0004   YMINLO RMB    4
116A 20            FCC    / (HEX)/
1170 0D0A          FDB    $0D0A
1172 20            FCC    / ZMAX= /
```

```
1179 0002   ZMAXNM RMB   2
117B 20            FCC   / (HEX) AT SAMPLE NUMBER /
[illegible]
1197 20            FCC   / (HEX)/
119D 0D0A          FDB   $0D0A
119F 20            FCC   / ZMIN= /
11A6 0002   ZMINNM RMB   2
11A8 20            FCC   / (HEX) AT SAMPLE NUMBER /
11C0 0004   ZMINLO RMB   4
11C4 20            FCC   / (HEX)/
11CA 0D0A          FDB   $0D0A,$0D0A
11CE 43            FCC   /COMPRESSION STATISTICS:/
11E5 0D0A          FDB   $0D0A
11E7 20            FCC   / NUMBER OF MEMORY BITS /
11FE 41            FCC   /AVAILABLE = /
120A 0006   AVABIT RMB   6
1210 20            FCC   / (HEX)/
1216 0D0A          FDB   $0D0A
1218 20            FCC   / NUMBER OF BITS AVAILABLE/
1231 20            FCC   / TO VAR LEN CODER = /
1245 0008   TCPBIT RMB   8
124D 20            FCC   / (HEX)/
1253 0D0A          FDB   $0D0A
1255 20            FCC   / TOTAL NUMBER OF DATA BITS/
126F 20            FCC   / STORED = /
1279 0008   TOTBIT RMB   8
1281 20            FCC   / (HEX)/
1287 0D0A          FDB   $0D0A
1289 20            FCC   / NUMBER OF BITS USED TO/
12A0 20            FCC   / STORE CHANNEL X = /
12B3 0006   XBITNM RMB   6
12B9 20            FCC   / (HEX)/
12BF 0D0A          FDB   $0D0A
12C1 20            FCC   / NUMBER OF BITS USED TO/
12D8 20            FCC   / STORE CHANNEL Y = /
12EB 0006   YBITNM RMB   6
12F1 20            FCC   / (HEX)/
12F7 0D0A          FDB   $0D0A
12F9 20            FCC   / NUMBER OF BITS USED TO/
1310 20            FCC   / STORE CHANNEL Z = /
1323 0006   ZBITNM RMB   6
1329 20            FCC   / (HEX)/
132F 0D0A          FDB   $0D0A
1331 20            FCC   / NUMBER OF BITS USED TO/
1348 20            FCC   / STORE TIME OR OTHER PARAM/
1362 45            FCC   /ETER = /
1369 0006   TBITNM RMB   6
136F 20            FCC   / (HEX)/
1375 0D0A          FDB   $0D0A
1377 43            FCC   /COMPRESSION RATIO = TOTAL /
1391 44            FCC   /DATA BITS STORED PER MEM /
13AA 42            FCC   /BITS AVAILABLE/
13B8 0D0A          FDB   $0D0A,$0D23
13BC 50            FCC   /PRESS RETURN =/
```

```
07590 13CA 04          FCB    4
07600            *
07610            *  END OF PRSTAT
07620            *
07630                  END
```

```
                 **************************************************
                 *
                 * [illegible] : NOCPRS
                 * AUTHOR :  CAPT. MEL TOWNSEND
                 * VERSION : 1.7
                 * VERSION DATE : 3 OCT 80
                 *
                 **************************************************
                 *
                 * OVERLAY DESCRIPTION
                 *
                 * THIS OVERLAY SAMPLES THE EKG DATA VIA THE
                 * A/D CONVERTERS AND STORES THE 8 BIT ROUNDED
                 * VALUES INTO MEMORY LOCATIONS 3C00-7FFF. THE
                 * THE X,Y,Z CHANNELS ARE SAMPLED AND STORED
                 * SEQUENTIALLY STARTING AT 3C00 AND WORKING UP.
                 *
                 **************************************************
                 **************************************************
                 *
                 * START OF NOCPRS
                 *
                 **************************************************
                 *
0100                    ORG     $0100     OVERLAY START ADDRESS
                 *
                        OPT     O         ASSB OPT-GEN OBJ FILE
                        OPT     NOG       ASSB OPT-SUPPRESS FULL FCC LI
                 *
                 **************************************************
                 *
                 * LABLE DECLARATIONS
                 *
                 * SUPPORT SUBROUTINE ADDRESSES
                 *
        CA87     OUTPUT EQU     $CA87     FPROMDOS. ALPH STRING TO CONS
        CA8F     OUTNCR EQU     $CA8F     FPROMDOS. ALPH STRING,NO CRLF
        CA2C     KEYED  EQU     $CA2C     FPROMDOS. CONSOLE INPUT NO TI
        CA36     KEYEDO EQU     $CA36     FPROMDOS. CONSOLE INPUT. NO ?
        1D00     START  EQU     $1D00     EKG-EXEC. START EKG-EXEC
                 *
                 * DATA BUFFLRS
                 *
        3400     HDRSTR EQU     $3400     DATA MEMORY BUFFER HEADER
        FFF8     IRQVEC EQU     $FFF8     INTERRUPT VECTOR [illegible]
        1C96     STKSAV EQU     $1C96     STACK SAVE [illegible]
        1C98     CPRTYP EQU     $1C98     COMPRESSION [illegible]
        3002     ENDBUF EQU     $3002     ADDR [illegible]
        1C9D     VECSAV EQU     $1C9D     IRQ VECTOR SAVE [illegible]
        1CA1     FILHLC EQU     $1CA1     FILHDR ADDR [illegible]
        1CA3     SAVHLC EQU     $1CA3     SAVHL ADDR [illegible]
        3490     LOOPCT EQU     $3490     TOTAL [illegible]
        3494     SAMPNO EQU     $3494     NUMBER OF [illegible]
        3496     LPCAL  EQU     $3496     AVG'D MAX LOOP [illegible]
```

```
        3497    MAXZ   EQU    $3497    MAX VLU IN CH Z
        3498    MAXZLO EQU    $3498    MAX VLU LOC IN CH Z
[illegible]
        349B    MINZLO EQU    $349B    MIN VLU LOC IN CH Z
        349D    MAXY   EQU    $349D    MAX VLU IN CH Y
        349E    MAXYLO EQU    $349E    MAX VLU LOC IN CH Y
        34A0    MINY   EQU    $34A0    MIN VLU IN CH Y
        34A1    MINYLO EQU    $34A1    MIN VLU LOC IN CH Y
        34A3    MAXX   EQU    $34A3    MAX VLU IN CH X
        34A4    MAXXLO EQU    $34A4    MAX VLUE LOC IN CH X
        34A6    MINX   EQU    $34A6    MIN VLU IN CH X
        34A7    MINXLO EQU    $34A7    MIN VLU LOC IN CH X
        34AC    DTABIT EQU    $34AC    NUM OF BITS USED TO STR DTA
        34B0    XBITS  EQU    $34B0    NUM OF BITS USED TO STR X
        34B3    YBITS  EQU    $34B3    NUM OF BITS USED TO STR Y
        34B6    ZBITS  EQU    $34B6    NUM OF BITS USED TO STR Z
        34B9    TBITS  EQU    $34B9    NUM OF BITS USED TO STR T
        34BC    ACHLCT EQU    $34BC    # BITS SET TO VLC IN CTR HECC
        3600    XPDFM  EQU    $3600    0 VLU LOC OF X PDF
        3800    YPDFM  EQU    $3800    0 VLU LOC OF Y PDF
        3A00    ZPDFM  EQU    $3A00    0 VLU LOC OF Z PDF
        3B00    TPDF   EQU    $3B00    0 VLU LOC TO TIME PDF
        3C00    SECZRO EQU    $3C00    START OF DATA MEM FILE
                *
                * HARDWARE ADDRESSES
                *
        E400    ADCZRO EQU    $E400    ADC CHANNEL ZERO
        E404    ADCTWO EQU    $E404    ADC CHANNEL TWO
        E500    DACZRO EQU    $E500    DAC CHANNEL ZERO
                *
                *
                *FUNCTION :NOCPRS
                *INPUTS :STATUS NUMBERS FROM EKG-EXEC
                *OUTPUTS :DATA TO DISK
                *CALLS  :OUTPUT,FILHDR,KEYBD,OUTNCR
                * KEYBD0,SAVFIL,START
                *DESTROYS :ALL REGISTERS
                *PURPOSE :TO COLLECT 3 CHANNELS OF EKG
                * DATA AND STORE INTO MEMORY.
                *
                *
0100 0F         NOCPRS SEI
0101 CE 0233           LDX    #STRNG
0104 BD CA8F           JSR    OUTNCR   "THIS MODULE SAMPLES THE ..."
0107 BD CA2C           JSR    KEYBD    GET RESPONSE FROM CONSOLE
010A FE 3002           LDX    ENDBUF
010D 09                DEX
010E E6 00             LDA B  0,X
0110 C1 59             CMP B  #'Y      IS INPUT YES?
0112 27 03             BEQ    NOCPR1   YES. KEEP EXECUTING THIS ROUT
0114 7E 1D00           JMP    START    NO. RTN TO EKG-EXEC
0117 CE 4E43    NOCPR1 LDX    #$4E43   FLAG COMPRESSION TYPE (NC)
011A FF 1C98           STX    CPRTYP
011D BD 051A           JSR    FILHDR   SET UP DATA FILE HEADER
```

```
01110 0120 CE 02B5          LDX    #IRQMSG
[illegible]                                  "[illegible]..."
[illegible]
01140 0129 CE 3C00 NOCPGO LDX    #SECZRO   INITIALIZE MEM BUF START LOC
01150 012C FF 0230          STX    BUFPTR    STORE IN BUFFER POINTER
01160 012F 86 03            LDA A  #3        PICK UP COUNTER FOR TIME CAL
01170 0131 B7 0227          STA A  CALCNT    STORE IN BUFFER
01180 0134 FE FFF8          LDX    IRQVEC    PICK UP CURRENT IRQ VECTOR
01190 0137 FF 1C9D          STX    VECSAV    SAVE IN BUFFER
01200 013A CE 051F TIMCAL LDX    #CALINT   GET INTR VECTOR ADDR FOR CAL
01210 013D FF FFF8          STX    IRQVEC    PUT IN VECTOR ADDRESS
01220 0140 86 00            LDA A  #0        INIT COUNTER FOR 256 TEST LOC
01230 0142 B7 0232          STA A  DONTST    STORE IN DONTST BUFFER
01240 0145 CE 0000          LDX    #0        CLEAR LOOP COUNTER
01250 0148 FF 022C          STX    IRQCNT
01260 014B FF 022E          STX    IRQCNT+2
01270 014E FF E500          STX    DACZRO    PULSE INT ENABLE CIRCUIT
01280 0151 20 09            BRA    SP001     BRA TO COUNTING LOOP
01290                  *
01300                  ****************************************************
01310                  *
01320                  * BASIC TIMING LOOP FOR EFFICIENCY TEST
01330                  *
01340                  ****************************************************
01350                  *
01360 0153 01          SPLOOP NOP              DELAY TO MATCH TIME
01370 0154 01                 NOP              IT TAKES TO EXECUTE THE
01380 0155 01                 NOP              INCREMENT OF BYTES 3&4
01390 0156 01                 NOP              OF THE LOOP COUNTER
01400 0157 01                 NOP              WHEN COUNT CARRIES TO
01410 0158 01                 NOP              HIGH 2 BYTES OF 4 BYTE
01420 0159 7E 015F            JMP    SP003     JUMP TO CONTINUE LOOP
01430 015C 0E          SP001  CLI              PREPARE FOR TIMER INT
01440 015D 3E                 WAI              STOP PROCESSOR & WAIT
01450 015E 0E          SP004  CLI
01460 015F FE 022E            LDX    IRQCNT+2 RET'ED FROM INT,INC COUNT
01470 0162 08                 INX
01480 0163 FF 022E            STX    IRQCNT+2 SAVE IT
01490 0166 26 EB              BNE    SPLOOP    COUNT GONE FFFF TO 0000?
01500 0168 FE 022C            LDX    IRQCNT    YES. INC BYTES 3&4
01510 016B 08                 INX
01520 016C FF 022C            STX    IRQCNT    SAVE COUNT
01530 016F B6 0232 SP003  LDA A  DONTST    IS DONE TEST SATISIFIED?
01540 0172 81 00              CMP A  #0
01550 0174 26 E8              BNE    SP004     NO. KEEP LOOPING
01560                  *
01570                  ****************************************************
01580                  *
01590                  * END BASIC TIMING LOOP FOR EFF TEST
01600                  *
01610                  ****************************************************
01620                  *
01630 0176 0F                 SEI              PREVENT SER TO NORL INTR
01640 0177 CE 4000            LDX    #$4000    PICK UP INTR OFF WORD
```

```
017A FF E500          STX     DACZRO    DISABLE INTR FLGH TIMER
017D B7 E400          STA A   ADCZRO    CLR MC6800 INT FLIP FLOP
[illegible]
[illegible] 7A [illegible]      [illegible]     CALC[illegible]  [illegible]
0184 27 38            BEQ     SPDONE    IS COUNT 3? YES, GO SAVE DATA
0186 B6 0227          LDA A   CALCNT    NO. GET COUNT
0189 81 02            CMP A   #2        IS COUNT 2?
018B 26 22            BNE     PARSAV    NO. SAVE PARAMETER FROM INPUT
018D FE 022E          LDX     IRQCNT+2 YES. SAVE TIME LOOP COUNTER
0190 FF 022A          STX     CALONE
0193 CE 0396          LDX     #SAMPLE   NOW PUT SAMPLE ADDR IN IRQVEC
0196 FF FFF8          STX     IRQVEC
0199 86 AA            LDA A   #$AA      SET UP DONTST
019B B7 0232          STA A   DONTST
019E CE 0000          LDX     #0        CLR LOOP COUNT AND SAMPLE COU
01A1 FF 022C          STX     IRQCNT
01A4 FF 022E          STX     IRQCNT+2
01A7 FF 3494          STX     SAMPNO
01AA FF E500          STX     DACZRO    ENABLE TIMER INTERRUPTS
01AD 20 AD            BRA     SPOO1     GO WAIT FOR INTERRUPT
01AF FE 022C PARSAV   LDX     IRQCNT    SAVE LOOP COUNT
01B2 FF 3490          STX     LOOPCT
01B5 FE 022E          LDX     IRQCNT+2
01B8 FF 3492          STX     LOOPCT+2
01BB 7E 013A          JMP     TIMCAL    GO EXECUTE ANOTHER TIME CAL
01BE B6 022B SPDONE   LDA A   CALONE+1 AVERAGE TWO TIME CAL RUNS
01C1 BB 022F          ADD A   IRQCNT+3 ADD LSB BYTES
01C4 B7 0229          STA A   CALZRO+1 STORE IN BUFFER
01C7 B6 022A          LDA A   CALONE    PICK UP LSB OF FIRST
01CA B9 022E          ADC A   IRQCNT+2 ADD WITH CARRY
01CD B7 0228          STA A   CALZRO    STORE IN BUFFER
01D0 77 0228          ASR     CALZRO    DIVIDE BY 2 TO AVERAGE
01D3 76 0229          ROR     CALZRO+1
01D6 86 08            LDA A   #8        SET UP COUNTER FOR /256
01D8 77 0228 CALSHF   ASR     CALZRO    NOW DIVIDE BY 256 FOR
01DB 76 0229          ROR     CALZRO+1 LOOPS/INTERRUPT
01DE 4A               DEC A
01DF 26 F7            BNE     CALSHF    8 SHIFTS YET?
01E1 B6 0229          LDA A   CALZRO+1 PICK UP SHIFTED RESULT
01E4 B7 3496          STA A   LPCAL     STORE IN FILE PAR BUFFER
01E7 B6 34AF          LDA A   DTABIT+3 PUT DATA BIT COUNT INTO ACBIC
01EA B7 34FF          STA A   ACBICT+3 SINCE NO COMPRESSION PERFORMED
01ED B6 34AE          LDA A   DTABIT+2
01F0 B7 34FE          STA A   ACBICT+2
01F3 B6 34AD          LDA A   DTABIT+1
01F6 B7 34FD          STA A   ACBICT+1
01F9 B6 34AC          LDA A   DTABIT
01FC B7 34FC          STA A   ACBICT
01FF FE 1C9D          LDX     VECSAV    YES. RESTORE IRQ VECTOR
0202 FF FFF8          STX     IRQVEC
0205 CE 0312          LDX     #SAVMSG
0208 BD CA8F          JSR     OUTMCR    "SAMPLING COMPLETE..."
020B BD C9EC          JSR     KEYPD     GET SAVE DECISION
020E FE 3002          LDX     ENDBUF
```

```
0211 09                 DEX
0212 E6 00              LDA B  0,X
0214 C1 59              CMP B  #'Y     IS DECISION YES?
0216 26 0C              BNE    EXJMP   NO. [illegible]
0218 BD 0515            JSR    SAVEFL  YES. SAVE THE FILE ON DISK
021B CE 0381            LDX    #SDONE
021E BD CA8F            JSR    OUTNCR
0221 BD CA2C            JSR    KEYBD
0224 7E 1D00 EXJMP      JMP    START   JUMP BACK TO EKG-EXEC
                  *
0227 0001    CALCNT RMB    1       TEST CAL LOOP COUNTER
0228 0002    CALZRO RMB    2       AVG'D CAL LOOPS DURING DAT CO
022A 0002    CALONE RMB    2       TEMP BUF FOR PRE COLLEC CAL
022C 0004    IRQCNT RMB    4       TEMP INTR LOOP COUNTER
0230 0002    BUFPTR RMB    2       BUFFER POINTER OF NEXT AVAIL
0232 0001    DONTST RMB    1       DONE TEST FLG FOR MEM FULL
                  *
0233 1A07    STRMSG FDB    $1A07
0235 54             FCC    /THIS MODULE SAMPLES THE EKG /
0251 49             FCC    /INPUT AND STORES THE DATA WITH/
026F 0D0A           FDB    $0D0A
0271 4E             FCC    /NO DATA COMPRESSION./
0285 0D0A           FDB    $0D0A,$0D0A
0289 44             FCC    /DO YOU WISH TO EXECUTE THIS /
02A5 4D             FCC    /MODULE (Y OR N)/
02B4 04             FCB    4
02B5 1A      IRQMSG FCB    $1A
02B6 49             FCC    /INSURE SUBJECT AND EKG DEVICE /
02D4 52             FCC    /READY!/
02DA 0D0A           FDB    $0D0A,$0D0A,$0D0A
02E0 50             FCC    /PRESS RETURN, THEN CLOSE INTERRUPT/
0302 20             FCC    / ENABLE SWITCH./
0311 04             FCB    4
0312 1A07    SAVMSG FDB    $1A07
0314 53             FCC    /SAMPLING COMPLETE.  PLEASE OPEN /
0334 49             FCC    /INTERRUPT ENABLE SWITCH./
034C 0D0A           FDB    $0D0A,$0D0A,$0D0A
0352 44             FCC    /DO YOU WISH TO SAVE THIS DATA ON /
0373 44             FCC    /DISK (Y OR N)/
0380 04             FCB    4
0381 20      SDONE  FCC    / PRESS RETURN/
038E 04             FCB    4
             *
               ***********************************************
               END NOCPRS
             ***********************************************
             *
             *
             *FUNCTION :SAMPLE
             *INPUTS  :STATUS BUFFERS
             *OUTPUTS :COMPRESSED,ROUNDED DATA IN MEM BUFF
             *CALLS   :NOTHING
             *PURPOSE :THIS ROUTINE SAMPLES THE EKG LEVELS,
             * ROUNDS THE VALUES TO 3 BITS (FROM 12)
```

```
                       * CALCULATES MAX,MIN,# OF BITS, SAMPLES
                       * ETC., AND SAVES THESE PARAM AND DATA
                       * TO [illegible]
                       *
                       *
038F 0001              SHFBUF RMB    1         TEMP SHIFT BUFFER
0390 0001              TEMPA1 RMB    1         TEMP REG, MSB CH 0
0391 0001              TEMPB1 RMB    1         TEMP REG, LSB CH 0
0392 0001              TEMPA2 RMB    1         TEMP REG, MSB CH 1
0393 0001              TEMPB2 RMB    1         TEMP REG, LSB CH 1
0394 0001              TEMPA3 RMB    1         TEMP REG, MSB CH 2
0395 0001              TEMPB3 RMB    1         TEMP REG, LSB CH 2
                       *
0396 FE 3494           SAMPLE LDX    SAMPNO    GET CUR SAMPL COUNT
0399 08                       INX
039A FF 3494                  STX    SAMPNO
039D BF 1C96                  STS    STKSAV    SAVE STACK PRT
03A0 8E 3280                  LDS    #$3280    INITIALIZE STACK IN UNUSED MEM
03A3 CE E400                  LDX    #ADCZRO   NOW PULSE A/D TO START CONV
03A6 A7 00                    STA A  0,X       FOR CHANNEL 0
03A8 01                       NOP
03A9 A6 00                    LDA A  0,X
03AB E6 01                    LDA B  1,X
03AD B7 0390                  STA A  TEMPA1
03B0 F7 0391                  STA B  TEMPB1
03B3 08                       INX
03B4 08                       INX
03B5 A7 00                    STA A  0,X       NOW PULSE A/D FOR CONV
03B7 01                       NOP              ON CHANNEL 1
03B8 A6 00                    LDA A  0,X
03BA E6 01                    LDA B  1,X
03BC B7 0392                  STA A  TEMPA2
03BF F7 0393                  STA B  TEMPB2
03C2 08                       INX
03C3 08                       INX
03C4 A7 00                    STA A  0,X       NOW PULSE A/D FOR CONV
03C6 01                       NOP              ON CHANNEL 2
03C7 A6 00                    LDA A  0,X
03C9 E6 01                    LDA B  1,X
03CB B7 0394                  STA A  TEMPA3
03CE F7 0395                  STA B  TEMPB3
03D1 CE 0390                  LDX    #TEMPA1
03D4 A6 00             SAMPL1 LDA A  0,X
03D6 E6 01                    LDA B  1,X
03D8 47                       ASR A            NOW SHIFT 2 BYTE VLU [illegible]
03D9 56                       ROR B            4 POSITIONS TO RIGHT [illegible]
03DA 47                       ASR A            12 BIT TO 8 BIT [illegible] CONV
03DB 56                       ROR B
03DC 47                       ASR A
03DD 56                       ROR B
03DE 47                       ASR A
03DF 56                       ROR B
03E0 F7 038F                  STA B  SHFBUF    SAVE 8 BIT RESULT OF SHIFT
03E3 86 00                    LDA A  #0        AND ADD CARRY OUT OF LAST
```

```
03F5 B9 039F        ADC A  SHFBUF    SHIFT ROUNDING UP OR DOWN
03F8 36             PSH A            SAVE VLU IN STACK TEMPORARILY
[illegible]         INX
[illegible]         INX
03FB 8C 0396        CPX    #TEMPA3+2 CHANNEL Z ROUNDED TO 8 BITS
03EE 26 E4          BNE    SAMPL1    NO. GO SAMPLE NEXT CHANNEL
03F0 32             PUL A            GET Z DATA
03F1 FE 0230        LDX    BUFPTR    PICK UP CUR MEM FILE PTR
03F4 A7 02          STA A  2,X       SAVE DATA TO MEM FILE
03F6 FE 3494        LDX    SAMPNO    GET CUR SAMPLE COUNT
03F9 B1 3497        CMP A  MAXZ      IS CUR Z MAX OVE SAMPLE SET?
03FC 2F 06          BLE    SPZMIN    NO. GO CHECK FOR MIN
03FE B7 3497        STA A  MAXZ      YES. KEEP CUR VLU
0401 FF 3498        STX    MAXZLO    KEEP CUR SAMPLE NUM
0404 B1 349A SPZMIN CMP A  MINZ      IS CUR Z MIN OVER SAMPLE SET?
0407 2C 06          BGE    SPYMAX    NO. GO CHECK FOR Y MAX
0409 B7 349A        STA A  MINZ      YES. KEEP CUR VLU
040C FF 349B        STX    MINZLO    KEEP CUR SAMPLE NUM
040F 32      SPYMAX PUL A            GET CUR Y DATA
0410 FE 0230        LDX    BUFPTR    PICK UP CURRENT MEM FILE PTR
0413 A7 01          STA A  1,X       SAVE DATA TO MEM FILE
0415 FE 3494        LDX    SAMPNO    GET CUR SAMPLE COUNT
0418 B1 349D        CMP A  MAXY      IS CUR Y MAX OVER SAMPLE SET
041B 2F 06          BLE    SPYMIN    NO. GO CHEK FOR MIN
041D B7 349D        STA A  MAXY      YES. KEEP CUR Y VLU
0420 FF 349E        STX    MAXYLO    KEEP CUR SAMPLE NUM
0423 B1 34A0 SPYMIN CMP A  MINY      IS CUR Y MIN OVER SAMPLE SET?
0426 2C 06          BGE    SPXMAX    NO. GO CHEK FOR X MAX
0428 B7 34A0        STA A  MINY      YES. KEEP CUR Y VLU
042B FF 34A1        STX    MINYLO    KEEP CUR SAMPLE NUM
042E 32      SPXMAX PUL A            GET CUR X DATA
042F FE 0230        LDX    BUFPTR    PICK UP CURRENT MEM FILE PTR
0432 A7 00          STA A  0,X       SAVE DATA TO MEM FILE
0434 FE 3494        LDX    SAMPNO    GET CUR SAMPLE COUNT
0437 B1 34A3        CMP A  MAXX      IS CUR X MAX OVER SAMPLE SET?
043A 2F 06          BLE    SPXMIN    NO. GO CHECK X MIN
043C B7 34A3        STA A  MAXX      YES. KEEP CUR X VLU
043F FF 34A4        STX    MAXXLO    KEEP CUR SAMPLE VLU
0442 B1 34A6 SPXMIN CMP A  MINX      IS CUR X MIN OVER SAMPLE SET?
0445 2C 06          BGE    MMDONE    NO. EXIT MAX,MIN UPDATE
0447 B7 34A6        STA A  MINX      YES. KEEP CUR X VLU
044A FF 34A7        STX    MINXLO    KEEP CUR SAMPLE COUNT
044D 0C      MMDONE CLC              DONE WITH MAX & MINS. NOW
044E 86 18          LDA A  #24       UPDATE ACTUAL DATA BIT STORED
0450 C6 00          LDA B  #0
0452 BB 34AF        ADD A  DTABIT+3
0455 B7 34AF        STA A  DTABIT+3
0458 86 00          LDA A  #0
045A F9 34AE        ADC B  DTABIT+2
045D F7 34AE        STA B  DTABIT+2
0460 C6 00          LDA B  #0
0462 B9 34AD        ADC A  DTABIT+1
0465 B7 34AD        STA A  DTABIT+1
0468 F9 34AC        ADC B  DTABIT
```

```
03810 046B F7 34AC          STA B  DTABIT
03820 046E 86 08            LDA A  #8        NOW UPDATE XBIT,YBIT,ZBIT COU
03830 0470 C6 00            [illegible]      [illegible]
03840 0472 BB 34B2          ADD A  [illegible] X,Y,Z [illegible]
03850 0475 B7 34B2          STA A  XBITS+2
03860 0478 B7 34B5          STA A  YBITS+2
03870 047B B7 34B8          STA A  ZBITS+2
03880 047E 86 00            LDA A  #0
03890 0480 F9 34B1          ADC B  XBITS+1
03900 0483 F7 34B1          STA B  XBITS+1
03910 0486 F7 34B4          STA B  YBITS+1
03920 0489 F7 34B7          STA B  ZBITS+1
03930 048C B9 34B0          ADC A  XBITS
03940 048F B7 34B0          STA A  XBITS
03950 0492 B7 34B3          STA A  YBITS
03960 0495 B7 34B6          STA A  ZBITS
03970 0498 7F 34BB          CLR    TBITS+2
03980 049B 7F 34BA          CLR    TBITS+1
03990 049E 7F 34B9          CLR    TBITS
04000 04A1 34               DES              NOW POINT STACK BACK TO DATA
04010 04A2 34               DES
04020 04A3 34               DES
04030 04A4 32               PUL A            GET Z DATA
04040 04A5 CE 3A00          LDX    #ZPDFM
04050 04A8 BD 04D2          JSR    PDFSTR    UPDATE Z PDF BIN COUNTER
04060 04AB 32               PUL A            GET Y DATA
04070 04AC CE 3800          LDX    #YPDFM
04080 04AF BD 04D2          JSR    PDFSTR    UPDATE Y PDF BIN CONTER
04090 04B2 32               PUL A
04100 04B3 CE 3600          LDX    #XPDFM
04110 04B6 BD 04D2          JSR    PDFSTR
04120 04B9 FE 0230          LDX    BUFPTR    NOW GET & UPDATE MEM BUFF PTR
04130 04BC 08               INX
04140 04BD 08               INX
04150 04BE 08               INX
04160 04BF FF 0230          STX    BUFPTR
04170 04C2 BE 1C96          LDS    STKSAV    RETRIEV ENTRY STACK POINTER
04180 04C5 8C 7FFE          CPX    #$7FFE    IS MEM BUF FULL?
04190 04C8 26 03            BNE    SAMRTI    NO. RET FROM INTR & SAMPLE AC
04200 04CA 7F 0232          CLR    DONTST    YES. RESET MEM FULL FLAG
04210 04CD 3B        SAMRTI RTI
04220                *
04230 04CE 0002      TEMPDF RMB    2         TEMP WORKING BUFFER
04240 04D0 0002      TEMPST RMB    2         TEMP STACK SAVE BUFFER
04250                *
04260 04D2 BF 04D0   PDFSTR STS    TEMPST    SAVE STACK PRT
04270 04D5 FF 04CE          STX    TEMPDF    SAVE PDF PTR
04280 04D8 16               TAB              SAVE INPUT DATA VLU
04290 04D9 BB 04CF          ADD A  TEMPDF+1  NOW CALCULATE ADDRS [illegible] INC
04300 04DC B7 04CF          STA A  TEMPDF+1  IN THE PDF MEM BUFF [illegible]
04310 04DF 86 00            LDA A  #0
04320 04E1 C5 80            BIT B  #$80      IS THE DATA VLU NEG?
04330 04E3 2B 05            BMI    PDFST1    YES. SBC MSB VS ADC
04340 04E5 B9 04CE          ADC A  TEMPDF    NO. ADD WITH CARRY MSB
```

```
04E8 20 03          BRA     PDFST2   AND STORE
04FA B2 04CE PDFST1 SBC A   TEMPDF   VLU IS NEG. SBC
0[illegible] B7 04CE [illegible] STA A [illegible]
04F0 17             TBA              [illegible]
04F1 BB 04CF        ADD A   TEMPDF+1 FOR PROPER ADDR CALCULATION
04F4 B7 04CF        STA A   TEMPDF+1
04F7 86 00          LDA A   #0
04F9 C5 80          BIT B   #$80     IS THE DATA NEG?
04FB 2B 05          BMI     PDFST3   YES. SBC MSB VS ADC
04FD B9 04CE        ADC A   TEMPDF   NO. ADD WITH CARRY MSB BYTE
0500 20 03          BRA     PDFST4   AND STORE
0502 B2 04CE PDFST3 SBC A   TEMPDF   VLU IS NEG. SUB WITH CARRY
0505 B7 04CE PDFST4 STA A   TEMPDF   AND STORE
0508 FE 04CE        LDX     TEMPDF   NOW LOAD CALC ADDR FOR INDEX
050B AE 00          LDS     0,X      GRAB VLU IN CALC ADDR
050D 31             INS              INCREMENT IT
050E AF 00          STS     0,X      AND STORE IT BACK IN BUFFER
0510 BE 04D0        LDS     TEMPST   RECOVER STACK POINTER
0513 17             TBA              RECOVER ORIGINAL DATA
0514 39             RTS
                    *
                    *FUNC: EXEC JUMPS
                    *INPUTS: ACCUMULATORS
                    *OUTPUTS: NONE
                    *CALLS: SAVEFL,FILHDR (EKG-EXEC) VIA ADDR BUFFERS
                    *DESTROYS: X,A,B,CC
                    *PURPOSE: TO JUMP TO DESIRED ROUTINES VIA RELOC ADDR
                    *
0515 FE 1CA3 SAVEFL LDX     SAVELC   GET ADDR OF SAVEFL
0518 6E 00          JMP     0,X      JUMP TO SAVEFL
                    *
051A FE 1CA1 FILHDR LDX     FILHLC   GET ADDR OF FILHLC
051D 6E 00          JMP     0,X      JUMP TO SAVEFL
                    *
                    ************************************************
                    * END SAMPLE
                    ************************************************
                    *
                    *
                    *FUNCTION :CALINT
                    *INPUTS (REG)  :NONE
                    *OUTPUTS (REG) :NONE
                    *CALLS  :NOTHING
                    *DESTROYS (REG):NONE (INTERRUPT HANDLER)
                    *PURPOSE :THIS ROUTINE IS USED FOR CALIBRATING
                    * THE MAX NUMBER OF LOOPS POSSIBLE
                    * DURING AN INTERRUPT PERIOD.  THIS
                    * ROUTINE UPDATES THE DONTST FLAG AND
                    * RESETS THE ST6800 INT FLIP FLOP &
                    * THE RETURNS.
                    *
051F 7C 0232 CALINT INC     DONTST   INCREMENT DONE TEST FLAG
0522 B7 E400        STA A   ADCZRO   CLR ST6800 INTR FLIP FLOP
0525 01             NOP
```

```
0526 3B          RTI
        *
        **************************************************
        *  END CALINT
        **************************************************
        *
        *
        **************************************************
        *
        *  END OF NOCPRS OVERLAY ROUTINES
        *
        **************************************************
        *
                 END
```

```
                  **************************************************
                  *
                  * OVERLAY NAME: TOLAN-A
                  * AUTHOR : CAPT. MEL TOWNSEND
                  * VERSION : 1.8
                  * VERSION DATE : 22 OCT 80
                  *
                  **************************************************
                  *
                  * OVERLAY DESCRIPTION
                  *
                  * THIS OVERLAY SAMPLES THE EKG DATA VIA THE
                  * A/D CONVERTERS, ROUNDS THE DATA TO 8 BITS,
                  * AND THEN COMPRESSES THE DATA VIA THE TOLAN-A
                  * ALGORITHM. THE COMPRESSED DATA IS THEN STORED INTO
                  * MEMORY DATA FILE FROM 3C00-7FFF.
                  *
                  **************************************************
                  *
                  * START OF TOLAN-A
                  *
                  **************************************************
                  *
0100                      ORG    $0100     OVERLAY START ADDRESS
                  *
                          OPT    O         ASSB OPT-GEN OBJ FILE
                          OPT    NOG       ASSB OPT-SUPPRESS FULL FCC LI
                  *
                  **************************************************
                  *
                  * LABLE DECLARATIONS
                  *
                  * SUPPORT SUBROUTINE ADDRESSES
                  *
        CA87      OUTPUT  EQU    $CA87     EPROMDOS. ALPH STRING TO CONS
        CA8F      OUTNCR  EQU    $CA8F     EPROMDOS. ALPH STRING,NO CRLF
        CA2C      KEYBD   EQU    $CA2C     EPROMDOS. CONSOLE INPUT ROUTI
        CA36      KEYBD0  EQU    $CA36     EPROMDOS. CONSOLE INPUT. NO ?
        1D00      START   EQU    $1D00     EKG-EXEC. FILE CREATE ROUTINE
                  *
                  * DATA BUFFERS
                  *
        3400      HDRSTR  EQU    $3400     DATA MEMORY BUFFER HEADER
        FFF8      IRQVEC  EQU    $FFF8     INTERRUPT VECTOR ADDR
        1C96      STKSAV  EQU    $1C96     STACK SAVE BUFFER
        1C98      CPRTYP  EQU    $1C98     COMPRESSION ALGOR FLAG
        3002      ENDBUF  EQU    $3002     ADDR OF LAST CHAR INPUT
        1C9D      VECSAV  EQU    $1C9D     IRQ VECTOR SAVE BUFFER
        1CA1      FILHLC  EQU    $1CA1     FILHDR ADDR PASS BUFFER
        1CA3      SAVFLC  EQU    $1CA3     SAVEFL ADDR PASS BUFFER
        3490      LOOPCT  EQU    $3490     TOTAL NUM TIME LOOP EXEC'D
        3494      SAMPNO  EQU    $3494     NUMBER OF SAMPLES TAKEN
        3496      LPCAL   EQU    $3496     AVG'D MAX LOOP COUNT/INTR
        3497      MAXZ    EQU    $3497     MAX VLU IN CH Z
```

```
          3498   MAXZLO EQU   $3498   MAX VLU LOC IN CH Z
          349A   MINZ   EQU   $349A   MIN VLU IN CH Z
          349B   MINZLO EQU   $349B   MIN VLU LOC IN CH Z
          349D   MAXY   EQU   $349D   MAX VLU IN CH Y
          349E   MAXYLO EQU   $349E   MAX VLU LOC IN CH Y
          34A0   MINY   EQU   $34A0   MIN VLU IN CH Y
          34A1   MINYLO EQU   $34A1   MIN VLU LOC IN CH Y
          34A3   MAXX   EQU   $34A3   MAX VLU IN CH X
          34A4   MAXXLO EQU   $34A4   MAX VLUE LOC IN CH X
          34A6   MINX   EQU   $34A6   MIN VLU IN CH X
          34A7   MINXLO EQU   $34A7   MIN VLU LOC IN CH X
          34AC   DTABIT EQU   $34AC   NUM OF BITS USED TO STR DTA
          34B0   XBITS  EQU   $34B0   NUM OF BITS USED TO STR X
          34B3   YBITS  EQU   $34B3   NUM OF BITS USED TO STR Y
          34B6   ZBITS  EQU   $34B6   NUM OF BITS USED TO STR Z
          34B9   TBITS  EQU   $34B9   NUM OF BITS USED TO STR T
          34BC   ACELCT EQU   $34BC   # BITS FED TO VAR LEN CODE
          3600   XPDFM  EQU   $3600   0 VLU LOC OF X PDF
          3800   YPDFM  EQU   $3800   0 VLU LOC OF Y PDF
          3A00   ZPDFM  EQU   $3A00   0 VLU LOC OF Z PDF
          3B00   TPDF   EQU   $3B00   0 VAL LOC OF TIME VAR HIST
          3C00   SECZRO EQU   $3C00   DATA STORE ADDR START
                 *
                 * HARDWARE ADDRESSES
                 *
          E400   ADCZRO EQU   $E400   ADC CHANNEL ZERO
          E404   ADCTWO EQU   $E404   ADC CHANNEL TWO
          E500   DACZRO EQU   $E500   DAC CHANNEL ZERO
                 *
                 *
                 *FUNCTION :TOLAN-A
                 *INPUTS :STATUS BUFFERS FROM EKG-EXEC
                 *OUTPUTS :DATA TO DISK
                 *CALLS :OUTPUT,FILHDR,KEYBD,OUTNCR
                 * KEYBD0,SAVEFL,START
                 *DESTROYS :ALL REGISTERS
                 *PURPOSE :TO COLLECT 3 CHANNELS OF EKG
                 * DATA AND STORE INTO MEMORY.
                 *
                 *
0100 0F          TOLANA SEI
0101 CE 0252            LDX    #STRMSG
0104 BD CA8F            JSR    OUTNCR    "THIS MODULE SAMPLES THE..."
0107 BD CA2C            JSR    KEYBD     GET RESPONSE FROM CONSOLE
010A FE 3002            LDX    ENDBUF
010D 09                 DEX
010E E6 00              LDA B  0,X
0110 C1 59              CMP B  #'Y       IS INPUT YES?
0112 27 03              BEQ    TOLAN1    YES. KEEP EXECUTING THIS ROUT
0114 7E 1D00            JMP    START     NO. RTN TO EKG-EXEC
0117 CE 5441     TOLAN1 LDX    #$5441    FLAG COMPRESSION TYPE (TA)
011A FF 1C98            STX    CPRTYP
011D BD 0238            JSR    FILHDR    SET UP DATA FILE HEADER
0120 CE 02D9            LDX    #IRQMSG
```

```
[illegible] C123 BD CA8F            JSR     OUTNCR   "INSURE SUBJECT AND EKG..."
[illegible] 0126 BD CA36            JSR     KEYPD0
[illegible]    9 CE 3C00 TCLAGO LDX     #SPCTRO  INITIALIZE [illegible]
[illegible] 012C FF 024E            STX     BUFPTR   SAVE IN BUFFER [illegible]
012F 86 04                 LDA A   #4       PICK UP COUNTER FOR TIME CAL
0131 B7 0245               STA A   CALCNT   STORE IN BUFFER
0134 86 80                 LDA A   #$80     INITIALIZE BIT POINTER TO LEF
0136 B7 03C7               STA A   BITPTR
0139 4F                    CLR A            INIT COMPRESS VAR
013A B7 0251               STA A   FIRST    CLEAR FIRST SAMPLE FLAG
013D B7 03BC               STA A   XSLOPE
0140 B7 03BD               STA A   YSLOPE
0143 B7 03BE               STA A   ZSLOPE
0146 FE FFF8               LDX     IRQVEC   PICK UP CURREN IRQ VECTOR
0149 FF 1C9D               STX     VECSAV   SAVE IN BUFFER
014C CE 06AE TIMCAL LDX     #CALINT  GET INTR VECTOR ADDR FOR CAL
014F FF FFF8               STX     IRQVEC   PUT IN VECTOR ADDRESS
0152 86 00                 LDA A   #0       INIT COUNTER FOR 256 TEST LOO
0154 B7 0250               STA A   DONTST   STORE IN DONTST BUFFER
0157 CE 0000               LDX     #0       CLEAR LOOP COUNTER
015A FF 024A               STX     IRQCNT
015D FF 024C               STX     IRQCNT+2
0160 FF E500               STX     DACZRO   PULSE INT ENABLE CIRCUIT
0163 20 09                 BRA     SP001    BRA TO COUNTING LOOP
                    *
                    ***********************************************
                    *
                    * BASIC TIMING LOOP FOR EFFICIENCY TEST
                    *
                    ***********************************************
                    *
0165 01             SPLOOP NOP              DELAY TO MATCH TIME
0166 01                    NOP              IT TAKES TO EXECUTE THE
0167 01                    NOP              INCREMENT OF BYTES 3&4
0168 01                    NOP              OF THE LOOP COUNTER
0169 01                    NOP              WHEN COUNT CARRIES TO
016A 01                    NOP              HIGH 2 BYTES OF 4 BYTE
016B 7E 0181               JMP     SP003    JUMP TO CONTINUE LOOP
016E 0E             SP001  CLI              PREPARE FOR TIMER INT
016F 3E                    WAI              STOP PROCESSOR & WAIT
0170 0E             SP004  CLI
0171 FE 024C               LDX     IRQCNT+2 RET'ED FROM INT,INC COUNT
0174 08                    INX
0175 FF 024C               STX     IRQCNT+2 SAVE IT
0178 26 EB                 BNE     SPLOOP   COUNT GONE FFFF TO 0000?
017A FE 024A               LDX     IRQCNT   YES. INC BYTES 3&4
017D 08                    INX
017E FF 024A               STX     IRQCNT   SAVE COUNT
0181 B6 0250 SP003  LDA A   DONTST   IS DONE TEST SATISFIED?
0184 81 00                 CMP A   #0
0186 26 E8                 BNE     SP004    NO. KEEP LOOPING
                    *
                    ***********************************************
                    *
```

```
                      * END BASIC TIMING LOOP FOR EFF TEST
                      *
                      ************************************************
                      *
0188 0F                      SEI                PREVENT SER TO MORE INTR
0189 CE 4000                 LDX     #$4000     PICK UP INTR OFF WORD
018C FF E500                 STX     DACZRO     DISABLE INTR FROM TIMER
018F B7 E400                 STA A   ADCZRO     CLR ST6800 INT FLIP FLOP
0192 01                      NOP
0193 7A 0245                 DEC     CALCNT     DEC TIME CAL LOOP COUNTER
0196 27 4F                   BEQ     SPDONE     IS COUNT 3? YES, GO SAVE DATA
0198 B6 0245                 LDA A   CALCNT     NO. GET COUNT
019B 81 03                   CMP A   #3         IS COUNT 3?
019D 26 25                   BNE     MAINSP     NO. GO TO MAIN SPL LOOP
019F FE 024C                 LDX     IRQCNT+2   YES. SAVE THE LOOP COUNTER
01A2 FF 0248                 STX     CALONE
01A5 CE 03C8                 LDX     #SAMPLE    NOW PUT SAMPLE ADDR IN IRQVEC
01A8 FF FFF8                 STX     IRQVEC
01AB 86 00                   LDA A   #$0        SET UP DONTST
01AD B7 0250                 STA A   DONTST
01B0 7F 0251                 CLR     FIRST      RESET FIRST SMP FLG
01B3 CE 0000                 LDX     #0         CLR LOOP COUNT AND SAMPLE COU
01B6 FF 024A                 STX     IRQCNT
01B9 FF 024C                 STX     IRQCNT+2
01BC FF 3494                 STX     SAMPNO
01BF FF E500                 STX     DACZRO     ENABLE TIMER INTERRUPTS
01C2 20 AA                   BRA     SP001      GO WAIT FOR INTERRUPT
01C4 81 02           MAINSP  CMP A   #2         IS COUNT 2?
01C6 26 10                   BNE     PARSAV     NO. SAVE PARAMETER FROM EKG
01C8 86 55                   LDA A   #$55       SET FIRST FLAG
01CA B7 0251                 STA A   FIRST
01CD B7 0250                 STA A   DONTST     SET DONTST FOR SMPL COLLEC
01D0 CE 0000                 LDX     #0         ENABLE INTR CLOCK
01D3 FF E500                 STX     DACZRO
01D6 20 96                   BRA     SP001      GO WAIT FOR INTERRUPT
01D8 FE 024A         PARSAV  LDX     IRQCNT     SAVE LOOP COUNT
01DB FF 3490                 STX     LOOPCT
01DE FE 024C                 LDX     IRQCNT+2
01E1 FF 3492                 STX     LOOPCT+2
01E4 7E 014C                 JMP     TIMCAL     GO EXECUTE ANOTHER TIME CAL
01E7 B6 0249         SPDONE  LDA A   CALONE+1   AVERAGE TWO TIME CAL RUNS
01EA BB 024D                 ADD A   IRQCNT+3   ADD LSB BYTES
01ED B7 0247                 STA A   CALZRO+1   STORE IN BUFFER
01F0 B6 0248                 LDA A   CALONE     PICK UP MSB OF FIRST
01F3 B9 024C                 ADC A   IRQCNT+2   ADD WITH CARRY
01F6 B7 0246                 STA A   CALZRO     STORE IN BUFFER
01F9 77 0246                 ASR     CALZRO     DIVIDE BY 2 TO AV
01FC 76 0247                 ROR     CALZRO+1
01FF 86 08                   LDA A   #8         SET UP COUNTER FOR 256
0201 77 0246         CALSHF  ASR     CALZRO     NOW DIVIDE BY 256 TO
0204 76 0247                 ROR     CALZRO+1   LOOPS/INTERRUPT
0207 4A                      DEC A
0208 26 F7                   BNE     CALSHF     8 SHIFTS YET?
020A B6 0247                 LDA A   CALZRO+1   PICK UP SHIFTED RESULT
```

```
020D B7 3496         STA A  LPCAL     STORE IN FILE HDR BUFFER
0210 BD 059F         JSR    DTACNT    NOW COUNT # OF DATA BYTES STOR
[illegible]
0216 FF FFF8         STX    IRQVEC
0219 CE 0336         LDX    #SAVMSG
021C BD CA8F         JSR    OUTNCR    "SAMPLING COMPLETE..."
021F BD CA2C         JSR    KEYBD     GET SAVE DECISION
0222 FE 3002         LDX    ENDBUF
0225 09              DEX
0226 E6 00           LDA B  0,X
0228 C1 59           CMP B  #'Y       IS DECISION YES?
022A 26 0C           BNE    EXJMP     NO. RTN TO EKG-EXEC
022C BD 0240         JSR    SAVEFL    YES. SAVE THE FILE ON DISK
022F CE 03A4         LDX    #SDONE
0232 BD CA8F         JSR    OUTNCR
0235 BD CA2C         JSR    KEYBD
0238 7E 1D00 EXJMP   JMP    START     JUMP BACK TO EKG-EXEC
                *
                *FUNC: RELOC JUMP
                *INPUTS: A
                *OUTPUTS: NONE
                *CALLS: FILHDR,SAVEFL (EKG-EXEC ROUTINES)
                *DESTROYS: A,B,CC,X
                *PURPOSE : THIS ROUTINE ENABLES TOLAN-A TO BE RELOCA
                *WITHOUT WORRY OF CHANGING CALLS TO EKG-EXEC.
                *
023B FE 1CA1 FILHDR LDX     FILHLC    GET FILHDR ADDRESS
023E 6E 00          JMP     0,X       JUMP TO IT
                *
0240 FE 1CA3 SAVEFL LDX     SAVELC    GET SAVEFL ADDRESS
0243 6E 00          JMP     0,X       JUMP TO IT
                *
0245 0001     CALCNT RMB    1         TEST CAL LOOP COUNTER
0246 0002     CALZRO RMB    2         AVG'D CAL LOOPS DURING DAT CO
0248 0002     CALCUR RMB    2         TEMP BUF FOR PRE COLLEC CAL
024A 0004     IRQCNT RMB    4         TEMP INTR LOOP COUNTER
024E 0002     BUTPTR RMB    2         BUFFER POINTER OF NEXT AVAIL
0250 0001     DONTST RMB    1         DONE TEST FLG FOR MEM FULL
0251 0001     FIRST  RMB    1         FIRST A/D SAMPL FLG
                *
0252 1A07     STRMSG FDB    $1A07
0254 54              FCC    /THIS MODULE SAMPLES THE EKG /
0270 49              FCC    /INPUT AND STORES THE DATA /
028A 0D0A            FDB    $0D0A
028C 43              FCC    /COMPRESSED VIA MAJOR TOLAN-A./
02A9 0D0A            FDB    $0D0A,$0D0A
02AD 44              FCC    /DO YOU WISH TO EXECUTE THIS
02C9 4D              FCC    /MODULE (Y OR N)/
02D8 04              FCB    4
02D9 1A       IRQMSG FCB    $1A
02DA 49              FCC    /INSURE SUBJECT AND EKG DEVIC
02F8 52              FCC    /READY!/
02FE 0D0A            FDB    $0D0A,$0D0A,$0D0A
0304 50              FCC    /PRESS RETURN, THEN CLOSE INTERRUPT/
```

```
0326 20            FCC    / ENABLE SWITCH./
0335 04            FCB    4
0336 1A07   SAMSWG FDB    $1A07
0338 53            FCC    /SAMPLING C... ... ...
0357 ..            FCC    /INTERRUPT ENABLE SWITCH./
036F 0D0A          FDB    $0D0A,$0D0A,$0D0A
0375 44            FCC    /DO YOU WISH TO SAVE THIS DATA ON /
0396 44            FCC    /DISK (Y OR N)/
03A3 04            FCB    4
03A4 20     SDONE  FCC    / PRESS RETURN/
03B1 04            FCB    4
            *
            *
            * END TOLAN-A
            *
            *
            *FUNCTION :SAMPLE
            *INPUTS :STATUS BUFFERS
            *OUTPUTS :COMPRESSED,ROUNDED DATA IN MEM BUFF
            *CALLS :NOTHING
            *PURPOSE :THIS ROUTINE SAMPLES THE EKG LEADS,
            * ROUNDS THE VALUES TO 8 BITS (FROM 12)
            * CALCULATES MAX,MIN,# OF BITS, SAMPLES
            * ETC., AND SAVES THESE PARAM AND DATA
            * TO DATA MEM FILE.
            *
            *
03B2 0001   SHFBUF RMB    1        TEMP SHIFT BUFFER
03B3 0001   TEMPA1 RMB    1        TEMP REG, MSB CH 0
03B4 0001   TEMPB1 RMB    1        TEMP REG, LSB CH 0
03B5 0001   TEMPA2 RMB    1        TEMP REG, MSB CH 1
03B6 0001   TEMPB2 RMB    1        TEMP REG, LSB CH 1
03B7 0001   TEMPA3 RMB    1        TEMP REG, MSB CH 2
03B8 0001   TEMPB3 RMB    1        TEMP REG, LSB CH 2
03B9 0001   XDATA  RMB    1        TEMP X CH DTA BUF
03BA 0001   YDATA  RMB    1        TEMP Y CH DTA BUF
03BB 0001   ZDATA  RMB    1        TEMP Z CH DTA BUF
03BC 0001   XSLOPE RMB    1        1ST DIFF X(N)-X(N+1)
03BD 0001   YSLOPE RMB    1        1ST DIFF Y(N)-Y(N+1)
03BE 0001   ZSLOPE RMB    1        1ST DIFF Z(N)-Z(N+1)
03BF 0001   XEXP   RMB    1        EXPECTED VAL OF X(N+1)
03C0 0001   YEXP   RMB    1        EXPECTED VAL OF Y(N+1)
03C1 0001   ZEXP   RMB    1        EXPECTED VAL OF Z(N+1)
03C2 0001   XACCEL RMB    1        DIFF X(N+1)-E(X(N+1))
03C3 0001   YACCEL RMB    1        DIFF Y(N+1)-E(Y(N+1))
03C4 0001   ZACCEL RMB    1        DIFF Z(N+1)-E(Z(N+1))
03C5 0001   STPFLG RMB    1        I HAVE JUST ... ...
03C6 0001   ICNT   RMB    1        TIME CNT VAR (<127)
03C7 0001   BITPTR RMB    1        BIT POINTER FOR SET & RESET
            *
03C8 FE 3494 SAMPLE LDX   SAMPNO   GET CUR SAMPL COUNT
03CB 08            INX
03CC FF 3494       STX    SAMPNO
03CF BF 1C96       STS    STKSAV   SAVE STACK PNT
```

```
03D2 8E 3280          LDS    #$3280   INITIALIZE STACK IN UNUSD MEM
03D5 B6 0251          LDA A  FIRST    CK FIRST FLG
[illegible]
03DA BD 05C8          JSR    SAMPLO   YES. SMPL & [illegible]
03DD FE 024E          LDX    BUFPTR   NOW STR FIRST DATA VLUES
03E0 F6 03F9          LDA A  XDATA
03E3 A7 00            STA A  0,X
03E5 08               INX
03E6 F6 03FA          LDA A  YDATA
03E9 A7 00            STA A  0,X
03EB 08               INX
03EC F6 03FB          LDA A  ZDATA
03EF A7 00            STA A  0,X
03F1 08               INX
03F2 FF 024E          STX    BUFPTR   UPDATE NEW BUFPTR
03F5 86 55            LDA A  #$55     SET STRFLG & FIRST FLAG
03F7 B7 03C5          STA A  STRFLG
03FA B7 0251          STA A  FIRST
03FD 86 08            LDA A  #8       UPDATE X,Y,Z BIT CNTRS FOR IN
03FF B7 34B2          STA A  XBITS+2
0402 B7 34B5          STA A  YBITS+2
0405 B7 34B8          STA A  ZBITS+2
0408 7F 0250          CLR    DONTST   CLEAR DONTST FLAG FOR 1 SMPL
040B BE 1C96          LDS    STKSAV
040E 3B               RTI
040F 7D 03C5 COMPRS   TST    STRFLG   HAS DATA JUST BEEN STORED ?
0412 27 05            BEQ    COMPR1   NO. KEEP COUNTING TIME
0414 86 01            LDA A  #1       YES. SET TIME COUNT TO 1
0416 F7 03C6          STA A  ICNT
0419 B6 03F9 COMPR1   LDA A  XDATA    GET X VLU
041C BB 03BC          ADD A  XSLOPE   ADD {X(N-1)-X(N-2)}
041F B7 03BF          STA A  XEXP     CREATE EXPECTED VAL OF X(N+1)
0422 F6 03FA          LDA A  YDATA    GET LAST Y SMPL VALUE
0425 BB 03BD          ADD A  YSLOPE   ADD {(Y(N-1)-Y(N-2)}
0428 B7 03C0          STA A  YEXP     CREATE EXPECTED VAL OF Y(N+1)
042B F6 03FB          LDA A  ZDATA    GET LAST Z SMPL VALUE
042E BB 03BE          ADD A  ZSLOPE   ADD {(Z(N-1)-Z(N-2)}
0431 B7 03C1          STA A  ZEXP     CREATE EXPECTED VAL OF Z(N+1)
0434 BD 05C8          JSR    SAMPLO   NOW GO GET NEXT SMPL. N=N+1
0437 F6 03F9          LDA A  XDATA    CALC DIF X(N)-E{X(N)}
043A B0 03BF          SUB A  XEXP
043D B7 03C2          STA A  XACCEL   SAVE DIF IN VAR
0440 B6 03FA          LDA A  YDATA    CALC DIF Y(N)-E{Y(N)}
0443 B0 03C0          SUB A  YEXP
0446 B7 03C3          STA A  YACCEL   SAVE DIF IN VAR
0449 F6 03FB          LDA A  ZDATA    CALC DIF Z(N)-E{Z(N)}
044C B0 03C1          SUB A  ZEXP
044F B7 03C4          STA A  ZACCEL   SAVE DIF IN VAR
0452 B6 03C2          LDA A  XACCEL
0455 26 21            BNE    COMPR2   ARE ALL EXTRAPOLATIONS GOOD?
0457 B6 03C3          LDA A  YACCEL   IF NO, NEED TO STR ALL DIFF'S
045A 26 1C            BNE    COMPR2
045C B6 03C2          LDA A  XACCEL
045F 26 17            BNE    COMPR2
```

```
0461 B6 03C6          LDA A  ICNT     UPDATE RUN TIME COUNTER
0464 4C               INC A
[illegible]           CMP A  [illegible]  [illegible]?
0467 2F 71            BLE    COMPR3   YES. C[illegible] & [illegible] IT
0469 B7 03C6          STA A  ICNT     NO. INC ICNT & RTI
046C 7F 03C5          CLR    STRFLG   RESET I HAVE JUST STRED FLG
046F 86 55            LDA A  #$55
0471 B7 0250          STA A  DONTST   KEEP DONE TEST SET
0474 BE 1C96          LDS    STKSAV   RETRIEV STACK
0477 3B               RTI
                *
0478 CE 34B0 COMPR2   LDX    #XBITS   UPDATE X BIT CNTR
047B B6 03C2          LDA A  XACCEL
047E BD 0568          JSR    BITCNT
0481 CE 3600          LDX    #XPDFM   UPDATE CH X FREQ OF OCCUR HIS
0484 B6 03C2          LDA A  XACCEL
0487 BD 066B          JSR    PDFSTR
048A B6 03C2          LDA A  XACCEL
048D B6 03C2          LDA A  XACCEL
0490 BB 03BC          ADD A  XSLOPE   CALC & SAVE X(N-1)-X(N-2)
0493 B7 03BC          STA A  XSLOPE
0496 CE 34B3          LDX    #YBITS   UPDATE Y BIT CNTR
0499 B6 03C3          LDA A  YACCEL
049C BD 0568          JSR    BITCNT
049F CE 3800          LDX    #YPDFM   UPDATE CH Y FREQ OF OCCUR HIS
04A2 B6 03C3          LDA A  YACCEL
04A5 BD 066B          JSR    PDFSTR
04A8 B6 03C3          LDA A  YACCEL
04AB B6 03C3          LDA A  YACCEL
04AE BB 03BD          ADD A  YSLOPE   CALC & SAVE Y(N-1)-Y(N-2)
04B1 B7 03BD          STA A  YSLOPE
04B4 CE 34B6          LDX    #ZBITS   UPDATE Z BIT CNTR
04B7 B6 03C4          LDA A  ZACCEL
04BA BD 0568          JSR    BITCNT
04BD CE [illegible]00 LDX    #ZPDFM   UPDATE CH Z FREQ OF OCCUR HIS
04C0 B6 03C4          LDA A  ZACCEL
04C3 BD 066B          JSR    PDFSTR
04C6 B6 03C4          LDA A  ZACCEL
04C9 B6 03C4          LDA A  ZACCEL
04CC BB 03BE          ADD A  ZSLOPE   CALC & SAVE Z(N-1)-Z(N-2)
04CF B7 03BE          STA A  ZSLOPE
04D2 CE 34B9          LDX    #TBITS   NOW UPDATE TIME BIT CNTR
04D5 86 06            LDA A  #6
04D7 BD 0568          JSR    BITCNT
04DA B6 03C2 COMPR3   LDA A  XACCEL   PICK UP XACCEL VAL
04DD BD 051B          JSR    COMSTR   GO CODE & STORE IT
04E0 B6 03C3          LDA A  YACCEL   PICK UP YACCEL VAL
04E3 BD 051B          JSR    COMSTR   GO CODE & STORE IT
04E6 B6 03C4          LDA A  ZACCEL   PICK UP ZACCEL VAL
04E9 BD 051B          JSR    COMSTR   GO CODE & STORE IT
04EC B6 03C6          LDA A  ICNT     NOW GET ICNT [illegible]
04EF CE 3F00          LDX    #TPDF    UPDATE TIME HISTO[illegible]
04F2 BD 066B          JSR    PDFSTR
04F5 BD 055A          JSR    RESET    PUT IN 0 [illegible]
```

AD-A100 799 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 6/5
ANALYSIS AND PERFORMANCE EVALUATION OF ELECTROCARDIOGRAM DATA C--ETC(U)
DEC 80 M D TOWNSEND
UNCLASSIFIED AFIT/GE/EE/80D-46 NL

3 OF 3
AD A
100799

END
DATE
FILMED
7-81
DTIC

```
04F8 79 03C6          ROL    ICNT     ROL 7 BIT ICNT TO MS POSITION
04FB 86 08            LDA A  #8       SET CNTER FOR TIME STORE LOOP
04FD 4A        COMPR4 DEC A           DECREMENT LOOP CNTR
04FE 27 0F            BEQ    COMPR6   TIME STRD YET?
0500 79 03C6          ROL    ICNT     NO. ROTATE TIME WORD ONCE LEF
0503 25 05            BCS    COMPR5   1 IN CARRY?
0505 BD 055A          JSR    RESET    NO. PUT 0 IN MEM FILE
0508 20 F3            BRA    COMPR4
050A BD 0534   COMPR5 JSR    SET      YES. PUT 1 IN MEM FILE
050D 20 EE            BRA    COMPR4
050F 86 55     COMPR6 LDA A  #$55
0511 B7 03C5          STA A  STRFLG   NOW SET "I HAVE JUST STORED"
0514 B7 0250          STA A  DONTST   SET DONTST FLAG
0517 BE 1C96          LDS    STKSAV
051A 3B               RTI
               *
051B BD 055A   COMSTR JSR    RESET    PUT 0 IN MEM FILE FOR DELIM
051E 4D               TST A           NOW TEST STATUS OF A
051F 2A 06            BPL    COMST1   IS IT +?
0521 40               NEG A           NO. GET 2'S COMP FOR MAG
0522 BD 0534          JSR    SET      NO. STORE 1 FOR SIGN BIT
0525 20 03            BRA    COMST2
0527 BD 055A   COMST1 JSR    RESET    YES. STORE 0 FOR SIGN BIT
052A 4D        COMST2 TST A           NOW CNT DOWN & STORE DATA
052B 27 06            BEQ    COMST3   CNT 0?
052D BD 0534          JSR    SET      NO. STORE 1 TO MEM FILE
0530 4A               DEC A           DEC CNTR
0531 20 F7            BRA    COMST2
0533 39        COMST3 RTS
               *
               *
               *FUNC: SET
               *INPUTS: BUFPTR,BITPTR
               *OUTPUTS: BIT SET PT'ED TO BY ABOVE
               *CALLS: NOTHING
               *DESTROYS: B,X,CC
               *PURPOSE: THIS ROUTINE SETS THE BIT POINTED TO
               * BY BITPTR & BUFPTR AND THEN UPDATES THESE COUNTERS
               *
0534 FE 024E   SET    LDX    BUFPTR   GET CUR MEM WORD FOR DATA STO
0537 F6 03C7          LDA B  BITPTR   GET BIT IN THAT WORD TO BE SE
053A EA 00            ORA B  0,X      SET THE BIT
053C E7 00            STA B  0,X      SAVE IN MEM FILE
053E 0C        SET0   CLC             CLR CARRY FOR PROPER LEFT ROT
053F 76 03C7          ROR    BITPTR   ROLL BITPTR ONCE RIGHT
0542 24 15            BCC    SETRTS   WAS BITPTR DOWN TO 1ST BIT?
0544 76 03C7          ROR    BITPTR   YES. IN BUFPTR $ SET BITPTR T
0547 08               INX
0548 FF 024E          STX    BUFPTR
054B F6 024E          LDA B  BUFPTR
054E C1 80            CMP B  #$80
0550 26 07            BNE    SETRTS
0552 7F 0250          CLR    DONTST   YES. STOP DATA COLLEC & RTI
0555 BE 1C96          LDS    STKSAV
```

```
0558 3B              RTI
0559 39       SETRTS RTS
              *
              *FUNC: RESET
              *INPUTS: BUFPTR,BITPTR
              *OUTPUTS: BIT RESET IN MEM FILE PT'ED TO BY ABOVE
              *CALLS: NOTHING
              *DESTROYS: B,X,CC
              *PURPOSE: THIS ROUTINE RESETS THE BIT IN THE MEMFILE
              * POINTED TO BY BUFPTR & BITPTR & UPDATES THESE COUN
              *
055A FE 024E RESET  LDX    BUFPTR
055D F6 03C7        LDA B  BITPTR
0560 53             COM B
0561 E4 00          AND B  0,X      RESET BIT IN MEM WORD
0563 E7 00          STA B  0,X      SAVE BIT IN MEM FILE
0565 7E 053E        JMP    SET0     UPDATE CNTR'S AND CK IF MEM F
              *
              *FUNC: BITCNT
              *INPUTS: ACCEL VLU IN A, BIT CNTR IN X
              *OUTPUTS: UPDATED BIT COUNT
              *CALLS:NOTHING
              *DESTROYS:A,B,CC
              *PURPOSE: THIS ROUTINE INC 3 BYTE BIT COUNTER WITH
              * NUMBER OF BIT USED TO STORE X,Y,Z,& TIME.
              *
0568 2A 01    BITCNT BPL   BITCN1   IS ACCEL +
056A 40             NEG A           NO. GET ABS VLU
056B 4C       BITCN1 INC A          ADD 1 FOR DELIM 0 BIT
056C 4C             INC A           ADD 1 FOR SIGN BIT
056D C6 00          LDA B  #0       RESET B WITHOUT AFFECTING CAR
056F AB 02          ADD A  2,X      INC LSB BYTE OF COUNT PTED TO
0571 A7 02          STA A  2,X
0573 86 00          LDA A  #0
0575 E9 01          ADC B  1,X
0577 E7 01          STA B  1,X
0579 C6 08          LDA B  #8       NOW UP COUNT OF ACTUAL DATA B
057B A9 00          ADC A  0,X
057D A7 00          STA A  0,X
057F 86 00          LDA A  #0
0581 FB 34BF        ADD B  ACELCT+3 NO. INC ACELCT COUNT
0584 F7 34BF        STA B  ACELCT+3
0587 C6 00          LDA B  #0
0589 B9 34BE        ADC A  ACELCT+2
058C B7 34BE        STA A  ACELCT+2
058F 86 00          LDA A  #0
0591 F9 34BD        ADC B  ACELCT+1
0594 F7 34BD        STA B  ACELCT+1
0597 B9 34BC        ADC A  ACELCT
059A B7 34BC        STA A  ACELCT
059D 39       BITRTS RTS
              *
              *FUNC: DTACNT
              *INPUTS: NUM ON SMPLS COLLECTED
```

```
                           *OUTPUTS: BITCNT=TO SMPLS*24
                           *CALLS: NOTHING
                           *DESTROYS: A,B,X,CC
                           *PURPOSE: TO CALC NUM OF BITS STORED DURING COLLECTI
                           *
      059E FE 3494 DTACNT LDX     SAMPNO
      05A1 86 18   DTACN1 LDA A   #24
      05A3 C6 00          LDA B   #0
      05A5 BB 34AF        ADD A   DTABIT+3
      05A8 B7 34AF        STA A   DTABIT+3
      05AB 86 00          LDA A   #0
      05AD F9 34AE        ADC B   DTABIT+2
      05B0 F7 34AE        STA B   DTABIT+2
      05B3 C6 00          LDA B   #0
      05B5 B9 34AD        ADC A   DTABIT+1
      05B8 B7 34AD        STA A   DTABIT+1
      05BB F9 34AC        ADC B   DTABIT
      05BE F7 34AC        STA B   DTABIT
      05C1 09             DEX
      05C2 8C 0000        CPX     #0
      05C5 26 DA          BNE     DTACN1
      05C7 39             RTS
                           *
                           *FUNC: SAMPL0
                           *INPUTS: DATA VIA A/D CONV
                           *OUTPUTS: ROUNDED DATA TO XDTA,YDTA,ZDTA
                           *CALLS:PDFSTR (UPDATE PDF HISTOGRAMS)
                           *DESTROYS:A,B,X,CC
                           *PURPOSE: THIS ROUTINES SAMPLES THE A/D AND THEN
                           * ROUNDS THE DATA TO 8 BITS FROM 12. THE DATA
                           * IS PLACED IN XDTA,YDTA,ZDTA AND THE COLLEC
                           * STAT ARE UPDATED.
                           *
      05C8 CE E400 SAMPL0 LDX     #ADCZRO  NOW PULSE A/D TO START CONV
      05CB A7 00          STA A   0,X      ON CHANNEL 0
      05CD 01             NOP
      05CE A6 00          LDA A   0,X
      05D0 E6 01          LDA B   1,X
      05D2 B7 03B3        STA A   TEMPA1
      05D5 F7 03B4        STA B   TEMPB1
      05D8 08             INX
      05D9 08             INX
      05DA A7 00          STA A   0,X      NOW PULSE A/D TO CONV
      05DC 01             NOP              ON CHANNEL 1
      05DD A6 00          LDA A   0,X
      05DF E6 01          LDA B   1,X
      05E1 B7 03B5        STA A   TEMPA2
      05E4 F7 03B6        STA B   TEMPB2
      05E7 08             INX
      05E8 08             INX
      05E9 A7 00          STA A   0,X      NOW PULSE A/D TO CONV
      05EB 01             NOP              ON CH 2
      05EC A6 00          LDA A   0,X
      05EE E6 01          LDA B   1,X
```

```
05970 05F0 B7 03B7          STA A  TEMPA3
05980 05F3 F7 03B8          STA B  TEMPB3
05990 05F6 CE 03B3          LDX    #TEMPA1
06000 05F9 A6 00     SAMPL1 LDA A  0,X
06010 05FB E6 01            LDA B  1,X
06020 05FD 47               ASR A            12 BIT TO 8 BIT ROUNDED CONV
06030 05FE 56               ROR B
06040 05FF 47               ASR A
06050 0600 56               ROR B
06060 0601 47               ASR A
06070 0602 56               ROR B
06080 0603 47               ASR A
06090 0604 56               ROR B
06100 0605 F7 03B2          STA B  SHFBUF    SAVE 8 BIT RESULT OF SHIFT
06110 0608 86 00            LDA A  #0        AND ADD CARRY OUT OF LAST
06120 060A B9 03B2          ADC A  SHFBUF    SHIFT ROUNDING UP OR DOWN
06130 060D 36               PSH A            SAVE TO MEM FILE BUFFER VIA S
06140 060E 08               INX
06150 060F 08               INX
06160 0610 8C 03B9          CPX    #TEMPA3+2  CHANNEL Z ROUNDED TO 8 BITS
06170 0613 26 E4            BNE    SAMPL1    NO. GO SAMPLE NEXT CHANNEL
06180 0615 32               PUL A            GET Z DATA
06190 0616 B7 03BB          STA A  ZDATA
06200 0619 FE 3494          LDX    SAMPNO    GET CUR SAMPLE COUNT
06210 061C B1 3497          CMP A  MAXZ      IS CUR Z MAX OVE SAMPLE SET?
06220 061F 2F 06            BLE    SPZMIN    NO. GO CHECK FOR MIN
06230 0621 B7 3497          STA A  MAXZ      YES. KEEP CUR VLU
06240 0624 FF 3498          STX    MAXZLO    KEEP CUR SAMPLE NUM
06250 0627 B1 349A SPZMIN   CMP A  MINZ      IS CUR Z MIN OVER SAMPLE SET?
06260 062A 2C 06            BGE    SPYMAX    NO. GO CHECK FOR Y MAX
06270 062C B7 349A          STA A  MINZ      YES. KEEP CUR VLU
06280 062F FF 349B          STX    MINZLO    KEEP CUR SAMPLE NUM
06290 0632 32      SPYMAX   PUL A            GET CUR Y DATA
06300 0633 B7 03BA          STA A  YDATA
06310 0636 B1 349D          CMP A  MAXY      IS CUR Y MAX OVER SAMPLE SET
06320 0639 2F 06            BLE    SPYMIN    NO. GO CHEK FOR MIN
06330 063B B7 349D          STA A  MAXY      YES. KEEP CUR Y VLU
06340 063E FF 349E          STX    MAXYLO    KEEP CUR SAMPLE NUM
06350 0641 B1 34A0 SPYMIN   CMP A  MINY      IS CUR Y MIN OVER SAMPLE SET?
06360 0644 2C 06            BGE    SPXMAX    NO. GO CHEK FOR X MAX
06370 0646 B7 34A0          STA A  MINY      YES. KEEP CUR Y VLU
06380 0649 FF 34A1          STX    MINYLO    KEEP CUR SAMPLE NUM
06390 064C 32      SPXMAX   PUL A            GET CUR X DATA
06400 064D B7 03B9          STA A  XDATA
06410 0650 B1 34A3          CMP A  MAXX      IS CUR X MAX OVER SAMPLE SET?
06420 0653 2F 06            BLE    SPXMIN    NO. GO CHECK X MIN
06430 0655 B7 34A3          STA A  MAXX      YES. KEEP CUR X VLU
06440 0658 FF 34A4          STX    MAXXLO    KEEP CUR SAMPLE VLU
06450 065B B1 34A6 SPXMIN   CMP A  MINX      IS CUR X MIN OVER SAMPLE SET?
06460 065E 2C 06            BGE    MMDONE    NO. EXIT MAX,MIN UPDATE
06470 0660 B7 34A6          STA A  MINX      YES. KEEP CUR X VLU
06480 0663 FF 34A7          STX    MINXLO    KEEP CUR SAMPLE COUNT
06490 0666 39      MMDONE   RTS
06500              *
```

```
0667 0002    TEMPDF RMB    2          TEMP WORKING BUFFER
0669 0002    TEMPST RMB    2          TEMP STACK SAVE BUFFER
             *
066B BF 0669 PDFSTR STS    TEMPST     SAVE STACK PTR
066E FF 0667        STX    TEMPDF     SAVE PDF PTR
0671 16             TAB
0672 BB 0668        ADD A  TEMPDF+1   NOW CALCULATE ADDRESS IN INC
0675 B7 0668        STA A  TEMPDF+1   IN THE PDF MEM BUFF
0678 86 00          LDA A  #0
067A C5 80          BIT B  #$80       IS THE DATA VLU NEG
067C 2B 05          BMI    PDFST1     YES. SBC MSB VS ADC
067E B9 0667        ADC A  TEMPDF     NO. ADC MSB
0681 20 03          BRA    PDFST2     AND STORE
0683 B2 0667 PDFST1 SBC A  TEMPDF     VLU IS NEG. SBC
0686 B7 0667 PDFST2 STA A  TEMPDF     AND STORE
0689 17             TBA               RECOVER VLU AND ADD (SUB) AGA
068A BB 0668        ADD A  TEMPDF+1   FOR PROPER ADDR CALC
068D B7 0668        STA A  TEMPDF+1
0690 86 00          LDA A  #0
0692 C5 80          BIT B  #$80       IS VLU NEG?
0694 2B 05          BMI    PDFST3     YES. SBC MSB VS ADC MSB
0696 B9 0667        ADC A  TEMPDF     NO. ADC MSB
0699 20 03          BRA    PDFST4
069B B2 0667 PDFST3 SBC A  TEMPDF     VLU IS NEG. SBC
069E B7 0667 PDFST4 STA A  TEMPDF     AND STORE
06A1 FE 0667        LDX    TEMPDF     NO LOAD CALC ADDR FOR INDEX
06A4 AE 00          LDS    0,X        GRAB VLU IN CALC ADDR
06A6 31             INS               INC IT
06A7 AF 00          STS    0,X        AND STORE IT BACK IN BUFFER
06A9 BE 0669        LDS    TEMPST     RECOVER STACK POINTER
06AC 17             TBA
06AD 39             RTS
             *
             *FUNC: CALINT
             *INPUTS: NONE
             *OUTPUTS: NONE
             *CALLS: NOTHING
             *DESTROYS :NO REG (INTR HANDLER)
             *PURPOSE: THIS ROUTINE IS USED FOR CALIBRATING
             * THE MAX NUMBER OF LOOPS POSSIBLE DURING
             * AN INTERRUPT PERIOD.  THIS ROUTINE UPDATES THE
             * DONTST FLAG AND RESETS THE ST6800 INT FLIP FLOP
             * AND THEN RETURNS.
             *
06AE 7C 0250 CALINT INC    DONTST     INCREMENT DONE TEST FLAG
06B1 B7 E400        STA A  ADCZRO     CLR ST6800 INTR FLIP FLOP
06B4 01             NOP
06B5 3B             RTI
             *
             * END OF TOLAN-A OVERLAY ROUTINES
             *
                    END
```

```
*
*       PROGRAM NAME:  DECPRS
*       AUTHOR :  CAPT. MEL TOWNSEND
*       VERSION : 1.3
*       VERSION DATE : 29 NOV 80
*
****************************************************
*
*        PROGRAM DESCRIPTION
*
*    THIS PROGRAM PERFORMS THE DECOMPRESSION OPERATI
* ON THE DATA COMPRESSED BY THE TOLAN-A,TOLAN-B,DOWE
* ETC.  THE ROUTINE ASSUMES THAT THE DATA IS IN MEMO
* AND READS THE COMPRESSION TYPE FROM THE MEMORY FIL
* BUFFER HEADER.  AFTER PROMPT TO THE TERMINAL, THIS
* PROGRAM BEGINS DECOMPRESSION AND OUTPUTS THE DECOM
* DATA TO AN ANALOG DISPLAY DEVICE (IE OSCILLISCOPE)
* D/A CONVERTER 0.  THE ROUTINE REQUIRES AN INTERRUP
* CLOCK VIA THE A/D BOARD (ST6800) AND THE ASSUMPTIO
* IS MADE THAT THE INTERRUPT CLOCK FREQUENCY IS ADJU
* TO THE SAMPLE RATE AT WHICH THE DATA WAS TAKEN.
*
****************************************************
*
*              START OF DECPRS
*
****************************************************
*
      0500          ORG     $0500     PROGRAM ORIGIN
*
                    OPT     O         ASSB OPT. LIST ASSMBLY
                    OPT     NOG       ASSB OPT. SUPRESS FCC LIST
*
****************************************************
*
*              LABLE DESCRIPTIONS
*
*  SUPPORT SUBROUTINES
*
      CA8F   OUTNCR EQU     $CA8F     EPROMDOS. OUTPUT STRING
      CA2C   KEYBDO EQU     $CA2C     EPROMDOS. INPUT ALPH STRING
      0100   DISPLA EQU     $0100     EKG-EXEC. TERMINAL INTFC DRIV
*
* DATA BUFFERS
*
      0020   CHNLBF EQU     $0020     CHNL SELECT FLG FRM DISPLAY
      3C00   XINIT  EQU     $3C00     INITIAL COND CH X
      3C01   YINIT  EQU     $3C01     INITIAL COND CH Y
      3C02   ZINIT  EQU     $3C02     INITIAL COND CH Z
      3002   ENDBUF EQU     $3002     END-OF-STRING FRM TERM INPUT
      FFF8   IRQVEC EQU     $FFF8     MASKABLE INTR JMP VECTOR ADDR
      3400   HDRSTR EQU     $3400     MEM FILE HEADER ADDR
      3C00   SECZRO EQU     $3C00     DATA START LOCATION
*
```

```
                * HARDWARE ADDRESSES
                *
           E500 DACZRO EQU    $E500    D/A CH 0
           E400 ADCZRO EQU    $E400    A/D CH 0
                *
                *
                *FUNCTION: DECPRS
                *INPUTS : CHNL SELECT VIA CHNLBF
                *OUTPUTS : DECRPRSED DATA VIA D/A
                *CALLS : OUTNCR,KEYBD0,DISPLAY
                *DESTROYS (REG) : A,B,X,CC
                *PURPOSE : THIS IS THE SET UP ROUTINE WHICH INITIALI
                * THE PROGRAM CONSTANTS AND, DETERMINES WHAT TYPE OF
                * COMPRESSION WAS USED, AND ENABLES RECONSTRUCTION
                * INTERUPPTS.
                *
0500 0F         DECPRS SEI
0501 FE 3400           LDX    HDRSTR   GET CMPRS TYPE FROM MEM FILE
0504 FF 0538           STX    NOTDN1   STR CPR TYPE IN ERROR MSG
0507 8C 5441           CPX    #$5441   IS IT TOLAN-A?
050A 27 61             BEQ    TADECP   YES. GO DECPRS IT.
050C CE 0518           LDX    #NOTDNE  NO. CPRS ROUTINE NOT DONE
050F BD CA8F           JSR    OUTNCR   PRINT ERROR MSG & RETURN
0512 BD CA2C           JSR    KEYBD0
0515 7E 0100           JMP    DISPLA   RETURN TO DISPLAY ROUTINE
                *
0518 1A07       NOTDNE FDB    $1A07,$0D0A
051C 44                FCC    /DECOMPRESSION FOR FILE TYPE /
0538 0002       NOTDN1 RMB    2
053A 20                FCC    / NOT YET IMPLEMENTED./
054F 0D0A              FDB    $0D0A,$0D0A
0553 50                FCC    /PRESS ANY KEY TO CONTINUE/
056C 04                FCB    4
                *
056D 96 20      TADECP LDA A  CHNLBF   GET WHICH CHANNEL TO BE DECPR
056F 81 00             CMP A  #0       IS IT CHNL X?
0571 26 07      TADEC0 BNE    TADEC1   NO. CHECK IF Y.
0573 C6 58             LDA B  #'X      PUT ASCII X IN ERROR MSG
0575 F7 0846           STA B  CHNASC
0578 20 10             BRA    TADEC3
057A 81 01      TADEC1 CMP A  #1       IS IT CH Y?
057C 26 07             BNE    TADEC2   NO. MUST BE Z.
057E C6 59             LDA B  #'Y      PUT ASCII Y IN ERROR MSG
0580 F7 0846           STA B  CHNASC
0583 20 05             BRA    TADEC3
0585 C6 5A      TADEC2 LDA B  #'Z      PUT ASCII Z IN ERROR MSG
0587 F7 0846           STA B  CHNASC
058A CE 0811    TADEC3 LDX    #GOMSG   GET PROMPT MSG LOC
058D BD CA8F           JSR    OUTNCR   NOW PRINT PROMPT
0590 BD CA2C           JSR    KEYBD0
0593 FE 3002           LDX    ENDBUF   GET ANSWER FROM TERMINAL
0596 09                DEX
0597 E6 00             LDA B  0,X
0599 C1 4E             CMP B  #'N      WAS IT NO?
```

```
059B 26 03              BNE    TADEC4   YES. EXIT ROUTINE
059D 7E 0100            JMP    DISPLA
05A0 CE 3C03   TADEC4   LDX    #SECNFO+3  POSITION POINTER TO DTA STR
05A3 FF 0611            STX    PUPTR
05A6 86 80              LDA A  #$80     INITIALIZE BITPTR TO LEFT BIT
05A8 B7 0613            STA A  BITPTR
05AB 7F 0617            CLR    XSLOPE   INIT SLOPES TO ZERO (2ND INIT
05AE 7F 0618            CLR    YSLOPE
05B1 7F 0619            CLR    ZSLOPE
05B4 B6 3C00            LDA A  XINIT    GET FIRST INIT COND
05B7 B7 0614            STA A  XDTA     & PUT IN DATA BUF
05BA B6 3C01            LDA A  YINIT
05BD B7 0615            STA A  YDTA
05C0 B6 3C02            LDA A  ZINIT
05C3 B7 0616            STA A  ZDTA
05C6 7F 0629            CLR    TARFLG   CLR DECPRSION FLG
05C9 7F 061A            CLR    DELT     INIT TIME COUNTER
05CC FE FFF8            LDX    IRQVEC   PICK UP CURRENT INT VEC
05CF FF 061D            STX    VECSAV   SAVE IT IN TEMP BUFFER
05D2 CE 062F            LDX    #TARECN  PUT NEW INT VEC IN
05D5 FF FFF8            STX    IRQVEC
05D8 86 AA              LDA A  #$AA     SET DONTST FLAG
05DA B7 062A            STA A  DONTST
05DD CE 0000            LDX    #0
05E0 FF E500            STX    DACZRO   SEND ENABLE PLS TO INT CIRCUI
               *
05E3 0E        TAINTR   CLI             ENABLE CPU FOR RECONS INTR'S
05E4 B6 062A            LDA A  DONTST   CHECK IF DECPRS DONE
05E7 81 00              CMP A  #0       DONE YET?
05E9 26 F8              BNE    TAINTR   NO. KEEP LOOPING
05EB 0F                 SEI             INSURE INTR DISABLE
05EC CE 0878            LDX    #GOAGIN  PRINT GO AGAIN MSG
05EF BD CA8F            JSR    OUTNCR   "ENTER CONTINUATION COMMAND"
05F2 BD CA2C            JSR    KEYBD0
05F5 FE 3002            LDX    ENDBUF   NOW GET COMMAND
05F8 09                 DEX
05F9 E6 00              LDA B  0,X
05FB 17                 TBA             PUT IT IN A
05FC 90 28              SUB A  $28      TRANSFORM X TO 0
05FE 97 20              STA A  CHNLBF   PUT IN CHNL FLAG
0600 F7 0846            STA B  CHNASC   PUT ASCII IN ERROR MSG
0603 C1 58              CMP B  #'X      WAS IT .LT. 'X?
0605 2D 07              BLT    TADEC5   YES. EXIT ROUTINE
0607 C1 5A              CMP B  #'Z      WAS CMMD .GT. 'Z ?
0609 2E 03              BGT    TADEC5   YES. EXIT ROUTINE
060B 7E 05A0            JMP    TADEC4   NO. GO EXECUTE DECPRS AGAIN
060E 7E 0100   TADEC5   JMP    DISPLA   RETURN TO DISPLAY
               *
               **************************************************
               *        END  DECPRS INIT ROUTINE
               **************************************************
               *
               *
               * PARAMETER BUFFER
```

```
                *
0611 0002       BUFPTR RMB      2         CURRENT MEM FILE WORD POINTER
0613 0001       BITPTR RMB      1         CURRENT BIT IN [illegible] POINTER
0614 0001       XDTA   RMB      1         CH X DATA BUF
0615 0001       YDTA   RMB      1         CH Y DATA BUG
0616 0001       ZDTA   RMB      1         CH Z DATA BUF
0617 0001       XSLOPE RMB      1         CH X 1ST DIFF BUFFER
0618 0001       YSLOPE RMB      1         CH Y 1ST DIFF BUF
0619 0001       ZSLOPE RMB      1         CH Z 1ST DIFF
061A 0001       DELT   RMB      1         TIME COMPRESSION RUN LEN BUFF
061B 0001       VALUE  RMB      1         DATA RET BUF FROM GETVLU
061C 0001       CNT    RMB      1         COUNTER IN GETTIM
061D 0002       VECSAV RMB      2         IRQ VECTOR SAVE BUF
061F 0001       BITVLU RMB      1         BIT SET/RESET FLAG FROM GETBI
0620 0001       SIGN   RMB      1         SIGN BIT ON DECODED DATA
0621 0001       XACCEL RMB      1         UNCODED CH X ACCEL DATA
0622 0001       XTEMP  RMB      1         TEMP BUF USED IN DECPRS
0623 0001       YACCEL RMB      1         UNCODED CH Y ACCEL DATA
0624 0001       YTEMP  RMB      1         TEMP BUF USED IN DECPRS
0625 0001       ZACCEL RMB      1         UNCODED CH Z ACCEL DATA
0626 0001       ZTEMP  RMB      1         TEMP BUF USED IN DECPRS
0627 0002       WORKBF RMB      2         TEMP WORKING BUFFER
0629 0001       TARFLG RMB      1         FLG USED IN DECPRS ALGORITHM
062A 0001       DCNTST RMB      1         FLG USED TO EXIT INTR LOOP
062B 0002       DABUF  RMB      2         D/A CONV OUTPUT BUFFER
062D 0002       STKSAV RMB      2         TEMP STACK SAVE BUFFER
                *
                *FUNCTION : TARECN
                *INPUTS : BUFFER VALUES
                *OUTPUTS : D/A VALUES VIA ST-6800
                *CALLS : GETVLU,GETTIM,OUTDA
                *DESTROYS       A,B,X,CC
                *PURPOSE : THIS IS THE INTERRUPT SERVICE ROUTINE WHI
                * PERFORMS RECONSTRUCTION ON THE TOLAN-A ENCODED DAT
                * FILE IN MEMORY.  THE DATA IS DECODED AND OUTPUT ON
                * D/A (ST6800) CH 0.
                *
                *****************************************************
                *
062F B7 E400    TARECN STA A  ADCZRO    ACK INTR. RESET INTR FF ON ST
0632 01                NOP
0633 BF 062D           STS    STKSAV    SAVE ENTRY STACK FOR RTI
0636 B6 0846           LDA A  CHNASC    CHECK CHNL SELECT, OUTPUT CUR
0639 81 58             CMP A  #'X       IS CHNNL SELECT X ?
063B 26 08             BNE    TAREC1    NO. CHK IF Y
063D B6 0614           LDA A  XDTA      YES. GET CUR XDTA VALUE.
0640 BD 0796           JSR    OUTDA     OUTPUT VALUE VIA D/A CH 0
0643 20 12             BRA    TARE22
0645 81 59      TAREC1 CMP A  #'Y       IS CHNNL SELECT Y ?
0647 26 08             BNE    TAREC2    NO. MUST BE Z
0649 B6 0615           LDA A  YDTA      GET CUR YDTA VALUE.
064C BD 0796           JSR    OUTDA     OUTPUT Y VALUE VIA D/A CH 0
064F 20 06             BRA    TARE22
0651 B6 0616    TAREC2 LDA A  ZDTA      GET CUR ZDTA VALUE.
```

```
02190 0654 BD 0796        JSR    OUTDA    OUTPUT Z VALUE VIA D/A CH 0
02200 0657 B6 0629 TARE22 LDA A  TARFLG   WAS LAST DATA PT EXTROPOLATED
02210 065A 81 00          CMP A  #0
02220 065C 26 33          BNE    TAREC3   YES. [illegible]
02230 065E 86 58          LDA A  #'X      PUT ASCII X IN GETVLU ERROR M
02240 0660 B7 0A3D        STA A  OVFLCH
02250 0663 BD 0700        JSR    GETVLU   NOW GO GET NEXT XACCEL VALUE
02260 0666 B6 061B        LDA A  VALUE    AND PUT XACCEL
02270 0669 B7 0621        STA A  XACCEL
02280 066C B7 0622        STA A  XTEMP    SAVE XACCEL IN TEMP BUFF
02290 066F 86 59          LDA A  #'Y      PUT 'Y IN GETVLU ERROR MSG
02300 0671 B7 0A3D        STA A  OVFLCH
02310 0674 BD 0700        JSR    GETVLU   GET YACCEL DTA
02320 0677 B6 061B        LDA A  VALUE
02330 067A B7 0623        STA A  YACCEL   PUT IN YACCEL VAR
02340 067D B7 0624        STA A  YTEMP    PUT IN YACCEL TEMP VAR
02350 0680 86 5A          LDA A  #'Z      PUT 'Z IN GETVLU ERROR MSG
02360 0682 B7 0A3D        STA A  OVFLCH
02370 0685 BD 0700        JSR    GETVLU   GET ZACCEL DATA
02380 0688 B7 0625        STA A  ZACCEL   PUT IN ZACCEL VAR
02390 068B B7 0626        STA A  ZTEMP    PUT IN ZACCEL TEMP VAR
02400 068E BD 075E        JSR    GETTIM   NOW GET TIME COPRESSION RUN L
02410 0691 7F 0629 TAREC3 CLR    TARFLG   RESET FLAGS
02420 0694 7A 061A        DEC    DELT     NOW CK IF RUN LEN ONLY 1
02430 0697 27 09          BEQ    TAREC4   YES. ACCEL VAR NOT ZERO
02440 0699 7F 0621        CLR    XACCEL   ACCEL WAS ZRO SO CLR X-Y-ZACC
02450 069C 7F 0623        CLR    YACCEL
02460 069F 7F 0625        CLR    ZACCEL
02470 06A2 B6 0617 TAREC4 LDA A  XSLOPE   NOW GET LAST SLOPE
02480 06A5 BB 0621        ADD A  XACCEL   & CALC NEW SLOPE WITH ACCEL
02490 06A8 B7 0617        STA A  XSLOPE
02500 06AB B6 0618        LDA A  YSLOPE
02510 06AE BB 0623        ADD A  YACCEL
02520 06B1 B7 0618        STA A  YSLOPE
02530 06B4 B6 0619        LDA A  ZSLOPE
02540 06B7 BB 0625        ADD A  ZACCEL
02550 06BA B7 0619        STA A  ZSLOPE
02560 06BD B6 0614        LDA A  XDTA     NOW CALC DATA VALUE USING SLO
02570 06C0 BB 0617        ADD A  XSLOPE
02580 06C3 B7 0614        STA A  XDTA
02590 06C6 B6 0615        LDA A  YDTA
02600 06C9 BB 0618        ADD A  YSLOPE
02610 06CC B7 0615        STA A  YDTA
02620 06CF B6 0616        LDA A  ZDTA
02650 06D8 B6 061A        LDA A  DELT     WAS RUN LEN .GT. 2 PTS ?
02660 06DB 27 22          BEQ    TARRTI   IF YES, DONT GET REA[illegible] FIL
02670 06DD 81 01          CMP A  #1       JUST EXTROPOLATE PO[illegible]IL
02680 06DF 27 07          BEQ    TAREC5   DELTA IS REDUCED TO 1
02690 06E1 86 37          LDA A  #55      SET TARFL1
02700 06E3 B7 0629        STA A  TARFLG
02710 06E6 20 17          BRA    TARRTI
02720 06E8 B6 0622 TAREC5 LDA A  XTEMP    RETRIEVE ACCEL VLUS PREVIOUSL
02730 06EB B7 0621        STA A  XACCEL
02740 06EE B6 0624        LDA A  YTEMP
```

```
06F1 B7 0623        STA A  YACCEL
06F4 B6 0626        LDA A  ZTEMP
06F7 B7 0625        STA A  ZACCEL
06FA 86 55          LDA A  #$55     NOW SET TARFL2 FLAG
06FC B7 0629        STA A  TARFLG
06FF 3B      TARRTI RTI
             *
             **************************************************
             *        END OF INTERRUPT HANDLER
             **************************************************
             *
             *                SUBROUTINES
             *
             *
             *FUNCTION : GETVLU
             *INPUTS : BUFPTR,BITPTR,SIGN BUFFERS
             *OUTPUTS :VALUE BUFFER
             *CALLS : GETBIT
             *DESTROYS : A,X,CC
             *PURPOSE : THIS ROUTINE SCANS MEMORY STARTING AT POI
             * DEFINED BY BUFPTR,BITPTR AND GETS ACCEL VALUE. BUF
             * AND BITPTR ARE UPDATED.
             *
             *
0700 7F 061B GETVLU CLR    VALUE    INSURE VALUE IS CLEAR ON STAR
0703 BD 07ED        JSR    GETBIT   GET DELIMITER BIT
0706 B6 061F        LDA A  BITVLU
0709 81 00          CMP A  #0       IS DELIM BIT =0 ?
070B 27 06          BEQ    GETVL1   YES. GET NEXT BIT
070D CE 097D        LDX    #SYNCMS  NO. SYNC ERROR HAS OCCURED. F
0710 7E 07F5        JMP    ERROR
0713 7F 0620 GETVL1 CLR    SIGN     INSURE SIGN CLR
0716 BD 07ED        JSR    GETBIT   GET SIGN BIT
0719 B6 061F        LDA A  BITVLU
071C 81 00          CMP A  #0       IS ACCEL POS?
071E 27 03          BEQ    GETVL2   YES. LEAVE SIGN CLR
0720 73 0620        COM    SIGN     NO. SET SIGN FLAG
0723 BD 07ED GETVL2 JSR    GETBIT   GET FIRST CODE BIT
0726 B6 061F        LDA A  BITVLU
0729 26 11          BNE    GETVL4
072B FE 0611 GETVL3 LDX    BUFPTR   END OF VALUE DECODE. BACK UP
072E 78 0613        ASL    BITPTR   FOR CORRECT POSITIONING FOR N
0731 24 2A          BCC    GETVL6   VALUE FETCH
0733 79 0613        ROL    BITPTR
0736 09             DEX
0737 FF 0611        STX    BUFPTR
073A 20 21          BRA    GETVL6
073C 7C 061B GETVL4 INC    VALUE    NOW PAST DELIM & SIGN. NOW CO
073F 2A 06          BPL    GETVL5   BITS. IS CODE WORD > 127 ?
0741 CE 09FB        LDX    #OVFLMS  YES. OVERFLOW HAS OCCURED. PR
0744 7E 07F5        JMP    ERROR
0747 BD 07ED GETVL5 JSR    GETBIT   NOW GET DATA VALUE CODE BITS
074A B6 061F        LDA A  BITVLU
074D 81 00          CMP A  #0       DELIM DETECTED YET?
```

```
074F 26 EB              BNE     GETVL4   NO. KEEP FETCHING BITS
0751 B6 0620            LDA A   SIGN     YES. GET SIGN AND ADJUST CNT
[illegible] 81 00       CMP A   #0
0756 27 [illegible]     BEQ     GETVL3   SIGN 0. VALUE [illegible]
0758 70 061B            NEG     VALUE    SIGN 1. TAKE 2'S COMP, ADJ PT
075B 20 CE              BRA     GETVL3
075D 39          GETVL6 RTS
                 *
                 ***************************************************
                 *            END OF SUBROUTINE GETVLU
                 ***************************************************
                 *
                 *
                 *FUNCTION : GETTIM
                 *INPUTS : BITVLU (CURR BIT VALUE PT'ED TO)
                 *OUTPUTS : DELT (CURRENT TIME COUNT)
                 *CALLS : GETBIT, ERROR
                 *DESTROYS : A,X,CC REGISTERS
                 *PURPOSE : THIS ROUTINE, WHEN CALLED AFTER 3 GETVLU
                 * RETURNS THE VALUE OF THE RUN LENGTH TIME COUNTER I
                 * VARIABLE DELT.
                 *
                 *
075E BD 07BD     GETTIM JSR     GETBIT   GET DELIM BIT
0761 B6 061F            LDA A   BITVLU
0764 81 00              CMP A   #0       IS BIT 0?
0766 27 06              BEQ     GETTI1   YES. PROPER DELIM. CONTINUE
0768 CE 09AA            LDX     #SYNCTI  NO. SYNC ERROR. PRINT & ERR O
076B 7E 07F5            JMP     ERROR
076E 7F 061A     GETTI1 CLR     DELT     INSURE TIME CNT INITIALLY CLR
0771 86 07              LDA A   #7       INIT CNT FOR 7 BIT TIME
0773 B7 061C            STA A   CNT
0776 BD 07BD     GETTI2 JSR     GETBIT   GET DELTA T BIT
0779 B6 061F            LDA A   BITVLU
077C BA 061A            ORA A   DELT     SET LSB BIT ACCORD TO T BIT
077F B7 061A            STA A   DELT
0782 78 061A            ASL     DELT     SHIFT DELT FOR NEXT BIT
0785 7A 061C            DEC     CNT      DONE WITH TIME YET?
0788 26 EC              BNE     GETTI2   NO. KEEP FETCHING TIME BITS
078A B6 061A            LDA A   DELT     YES. GET TIME VALUE
078D 26 06              BNE     GETTI3   IS TIME CNT 0 ?
078F CE 0A3F            LDX     #TIMERR  YES. ERROR. PRNT ERR MSG
0792 7E 07F5            JMP     ERROR    AND ERROR OFF
0795 39          GETTI3 RTS
                 *
                 ***************************************************
                 *            END GETTIM SUBROUTINE
                 ***************************************************
                 *
                 *
                 *FUNCTION : OUTDA
                 *INPUTS : DATA VALUE TO BE D/A'ED IN ACC A
                 *OUTPUTS : VALUE IN ACC A VIA D/A CH 0
                 *CALLS : NOTHING
```

```
                        *DESTROYS : B,CC
                        *PURPOSE : THIS ROUTINE OUTPUTS THE DATA IN
                        * ACCUMULATOR A TO THE D/A FOR CONVERSION AND
                        * TRANSMISSION.  THE 8 BIT VALUE IS SHIFTED
                        * TO FORM THE 12 BIT OPERAND REQUIRED BY THE
                        * D/A (ST6800).
                        *
                        *
0796 36                 OUTDA   PSH A               SAVE VLU IN A ACC
0797 B7 062C                    STA A   DABUF+1     PUT VALUE IN D/A OUT BUFF
079A 7F 062B                    CLR     DABUF       CLR MSB BYTE OF D/A OUT BUFF
079D 78 062C                    ASL     DABUF+1     LEFT SHIFT 2 BYTE D/A WORD
07A0 79 062B                    ROL     DABUF       FOR CONVERSION OF 8 BIT
07A3 78 062C                    ASL     DABUF+1     VALUE TO 12 BIT
07A6 79 062B                    ROL     DABUF
07A9 78 062C                    ASL     DABUF+1
07AC 79 062B                    ROL     DABUF
07AF 78 062C                    ASL     DABUF+1
07B2 79 062B                    ROL     DABUF
07B5 FE 062B                    LDX     DABUF       NOW LD INDEX WITH 12 BIT VLU
07B8 FF E500                    STX     DACZRO      AND OUTPUT TO D/A
07BB 32                         PUL A               RETRIEV ENTY A ACC VLUE
07BC 39                         RTS
                        *
                        **************************************************
                        *        END SUBROUTINE OUTDA
                        **************************************************
                        *
                        *
                        *FUNCTION : GETBIT
                        *INPUTS : BUFPTR,BITPTR POINTERS
                        *OUTPUTS : BITVLU, DONTST FLAGS
                        *CALLS :NOTHING
                        *DESTROYS :A,CC
                        *PURPOSE : THIS ROUTINE CHECKS THE STATE OF THE BIT
                        * POINTED TO BY THE BUFPTR,BITPTR PAIR AND SETS THE
                        * THE BITVLU FLAG ACCORDINGLY.  THIS ROUTINE CHECK
                        * FOR END OF MEMORY AND FORCES EXIT OF RECONSTRUCTIO
                        * ROUTINES IF EOM IS DETECTED.
                        *
                        *
07BD 7F 061F GETBIT CLR     BITVLU      INSURE BITVLU STARTS RESET
07C0 FE 0611        LDX     BUFPTR      GET MEM WORD POINTER
07C3 F6 0613        LDA B   BITPTR      GET BIT POINTER IN THAT MEM W
07C6 E5 00          BIT B   0,X         CHECK IF BIT SET
07C8 27 03          BEQ     GETBI1      NO IT WAS RESET. RESET BITVLU
07CA 7C 061F        INC     BITVLU      YES. BIT WAS SET. SET BITVLU
07CD 74 0613 GETBI1 LSR     BITPTR      NOW UPDATE POINTERS
07D0 24 22          BCC     GETRTS
07D2 76 0613        ROR     BITPTR
07D5 08             INX
07D6 FF 0611        STX     BUFPTR
07D9 F6 0611        LDA B   BUFPTR      CHECK FOR END OF MEMORY
07DC C1 80          CMP B   #$80
```

```
07DE 26 14           BNE    GETRTS   NO. CONTINUE
07E0 0F              SEI             NOW TURN OFF INTR AND RET
07E1 CE 4000         LDX    #$4000
07E4 FF E500         STX    DACZRO
07E7 FE 061D         LDX    VECSAV
07EA FF FFF8         STX    IRQVEC
07ED 7F 062A         CLR    DONTST   YES. CLR DONTST FLAG AND RTI
07F0 BE 062D         LDS    STKSAV   RETRIEVE INTR STACK PTR
07F3 3B              RTI
07F4 39       GETRTS RTS
              *
              ***************************************************
              *            END OF SUBROUTINE GETBIT
              ***************************************************
              *
              *
              *FUNCTION : ERROR
              *INPUTS : MESSAGE STRING POINTED TO BY INDX REG
              *OUTPUTS : ERROR MESSAGE TO CONSOLE DEVICE
              *CALLS : OUTNCR,KEYBDO,DISPLA
              *DESTROYS : A,B,X,CC
              *PURPOSE : THIS ROUTINE PRINTS ERROR MESSAGES UPON
              * ERROR DETECTION BY THE DECOMPRESSION ROUTINES.
              *
              *
07F5 BD CA8F  ERROR  JSR    OUTNCR
07F8 CE 09DF         LDX    #ANYKEY  GET ANYKEY MESSAGE
07FB BD CA8F         JSR    OUTNCR   PRINT "TO CONTINUE PRESS ANY
07FE BD CA2C         JSR    KEYBDO
0801 0F              SEI             NOW CLR INTR CONDITION
0802 CE 4000         LDX    #$4000   AND RETURN TO DISPLAY
0805 FF E500         STX    DACZRO
0808 FE 061D         LDX    VECSAV   RETRIEV IRQ VEC ADDR
080B FF FFF8         STX    IRQVEC
080E 7E 0100         JMP    DISPLA   RETURN TO "DISPLAY"
              *
              ***************************************************
              *        END ERROR HANDLING ROUTINE
              ***************************************************
              *
              *         MESSAGE STRINGS
              *
0811 1A0D     GOMSG  FDB    $1A0D,$0A07
0815 44              FCC    /DECOMPRESSION OVERLAY/
082A 0D0A            FDB    $0D0A,$0D0A
082E 43              FCC    /CURRENT CHANNEL FLAG IS /
0846 0001     CHNASC RMB    1
0847 0D0A            FDB    $0D0A,$0D0A
084B 44              FCC    /DO YOU WISH TO EXECUTE THIS MODULE (
0877 04              FCB    4
0878 0D0A     GOAGIN FDB    $0D0A,$0A07
087C 44              FCC    /DECOMPRESSION AND DISPLAY COMPLETE/
089E 0D0A            FDB    $0D0A,$0D0A
08A2 45              FCC    /ENTER CONTINUATION COMMAND/
```

```
08BC 0D0A            FDB    $0D0A
08BE 20              FCC    /   X=DISPLAY CHANNEL X ON D-A CHANNE
08E5 0D0A            FDB    $0D0A
08E7 20              FCC    /   Y=DISPLAY CHANNEL Y ON D-A CHANNE
090E 0D0A            FDB    $0D0A
0910 20              FCC    /   Z=DISPLAY CHANNEL Z ON D-A CHANNE
0937 0D0A            FDB    $0D0A,$0D0A
093B 41              FCC    /ANY OTHER KEY RETURNS CONTROL TO "DI
0966 0D0A            FDB    $0D0A,$0D0A
096A 45              FCC    /ENTER COMMAND NOW=/
097C 04              FCB    4
097D 1A07     SYNCMS FDB    $1A07,$0D0A
0981 53              FCC    /SYNC ERROR DETECTED BY TOLAN-A DECOD
09A7 0D0A            FDB    $0D0A
09A9 04              FCB    4
09AA 1A07     SYNCTI FDB    $1A07,$0D0A
09AE 53              FCC    /SYNC ERROR DETECTED BY /
09C5 54              FCC    /TOLAN-A DURING TIME FETCH/
09DE 04              FCB    4
09DF 0D0A     ANYKEY FDB    $0D0A
09E1 54              FCC    /TO CONTINUE PRESS ANY KEY/
09FA 04              FCB    4
09FB 1A07     OVFLMS FDB    $1A07,$0D0A
09FF 41              FCC    /ACCEL VALUE OVERFLOW DETECTED BY TOL
0A2F 0D0A            FDB    $0D0A
0A31 20              FCC    / IN CHANNEL /
0A3D 0001     OVFLCH RMB    1
0A3E 04              FCB    4
0A3F 1A07     TIMERR FDB    $1A07,$0D0A
0A43 54              FCC    /TIME COUNT ERROR DETECTED BY TOLAN-A
0A6F 0D0A            FDB    $0D0A
0A71 54              FCC    /TIME CNT (DELT)=0/
0A82 04              FCB    4
              *
              ***************************************************
              *
              *      END OF DECPRS OVERLAY ROUTINES
              *
              ***************************************************
              *
                     END
```

```
0100 REM ****************************************
0110 REM *
0120 REM *       EKG ENTROPY CALCULATION PROGRAM
0130 REM *
0140 REM ****************************************
0150 REM *
0160 REM *    THIS PROGRAM READS THE FREQUENCY
0170 REM * OF OCCURENCE DATA IN MEMORY FROM AN
0180 REM * EKG DATA COLLECTION AND CALCULATES
0190 REM * THE ENTROPY OF THE X,Y,AND Z DATA
0200 REM * SOURCES.  THE CALCULATED ENTROPY IS
0210 REM * STORED BACK IN THE MEMORY FILE HEADER
0220 REM * IN ASCII. THIS HEADER CAN THEN BE
0230 REM * INSERTED INTO THE DISK FILE USING
0240 REM * MINIDOS.
0250 REM *
0260 REM ****************************************
0270 REM
0280 DIM D(255),N$(8),T(255),M9(6),T$(3)
0290 STRING= 8
0291 DIGITS= 4
0292  T$(1)="X"
0293  T$(2)="Y"
0294  T$(3)="Z"
0295 LINE= 80
0296  L7=6
0297  E1=13507
0313 REM
0325 REM ***********************************************
0326 REM * NOW GET FILENAME FROM MEMORY & PRINT IT
0327 REM ***********************************************
0328 REM
0330 FOR I=1 TO 8
0340  J=13313+I
0350  N$(I)=CHR$(PEEK(J))
0360  A$=A$+N$(I)
0370 NEXT I
0372  P$=CHR$(12)
0373 PRINT P$
0374 PRINT
0375 PRINT
0376 PRINT
0377 PRINT
0378 PRINT
0380 PRINT
0385 PRINT "EKG ENTROPY CALCULATION"
0386 PRINT
0387 PRINT
0390 PRINT "FILENAME.............";A$
0400 PRINT
0401 PRINT
0402 GOSUB 1696
0403 GOSUB 1900
0404 IF G$ = "S" THEN GO TO 1120
```

```
C410 REM
0411 REM ************************************************
0412 REM * B1=MEM FILE DESTINATION FOR ASCII RESULTS
0413 REM * 13856=$3620 (HEX)=FLAG MEM USED BY BASIC. DATA REPLACED
0414 REM * 13567=34FF (HEX)=XPDF BUFFER-1
0415 REM * 14079=36FF (HEX)=YPDF BUFFER-1
0416 REM * 14591=38FF (HEX)=ZPDF BUFFER-1
0417 REM * 13500=34BC (HEX)=BASSAV. MEM BUF FOR $3620 DATA SAVE
0419 REM ************************************************
0420 REM
0421 REM
0424 REM
0425 REM ************************************************
0426 REM *  NOW GET VLU IN BASSAV & STR TC $3620 (HEX)
0427 REM ************************************************
0428 REM
0429  B=13856
0430  C1=1
0432  N=PEEK(13500)
0433 POKE( B,N)
0435 FOR I=1 TO 255
0436  T(I)=0
0437 NEXT I
0438  T9=0
0439  S2=0
0440  S=0
0441 REM
0442 REM ************************************************
0443 REM *  NOW BEGIN LOOP TO CALC X,Y,Z & TOTAL ENTROPY
0444 REM ************************************************
0445 REM
0450 IF C1 <> 1 THEN GO TO 480
0460  K=13567
0480 IF C1 <> 2 THEN GO TO 510
0490  K=14079
0510 IF C1 <> 3 THEN GO TO 540
0520  K=14591
0540 IF C1>= 4 THEN GO TO 800
0542 REM
0543 REM ****************************************************
0544 REM * GET 2 BYTE PDF DATA & MERGE INTO FLTING POINT NUMBER
0545 REM ****************************************************
0546 REM
0550 FOR I=1 TO 509 STEP 2
0560  M=(I+1)/2
0570  D(M)=256*PEEK(I+K)+PEEK(I+K+1)
0572  T(M)=T(M)+D(M)
0590  S=D(M)+S
0592  S2=D(M)+S2
0600 NEXT I
0605  D9=256*PEEK(K+511)+PEEK(K+512)
0606  S=S+D9
0607  T9=T9+D9
0608  S2=S2+D9
```

```
0610  E=0
0612 REM
0613 REM *****************************************************
0614 REM * NOW CALCULATE PROBABILITY OF OCCURENCE OF SPECIFIC
0615 REM * VLU'S FROM NUM OF OCCURENCES. THEN CALC ENTROPY SUM.
0616 REM *****************************************************
0617 REM
0620 FOR I=1 TO 255
0630  D(I)=D(I)/S
0640 IF D(I)=0 THEN GO TO 660
0650  E=E+(D(I)*(-(LOG(D(I))/.693147)))
0660 NEXT I
0663  D9=D9/S
0664 IF D9=0 THEN GO TO 666
0665  E=E+(D9*(-(LOG(D9)/.693147)))
0666 IF E=0 THEN GOTO 672
0667  C2=8/E
0668 GOTO 683
0672  C2=1
0673 GOTO 680
0674 REM
0675 REM *****************************************************
0676 REM *  PRINT RESULTS OF CALC TO TERMINAL (PRINTER).
0677 REM *****************************************************
0678 REM
0680 PRINT
0681 PRINT "MAXIMUM COMPRESSION FOR CHANNEL ";T$(C1);" IS INFINITE"
0682 GOTO 690
0683 PRINT
0684 PRINT "MAX COMPRESSION RATIO FOR CHANNEL ";T$(C1);" IS ";C2;" ; 1"
0690 PRINT
0700 PRINT "EKG LEAD ";T$(C1);" ENTROPY = ";E;" BITS."
0710 PRINT
0712 REM
0713 REM ********************************************************
0714 REM *  NOW CONVERT CALCULATED ENTROPY TO ASCII & STORE BACK
0715 REM *  IN MEMORY BUFFER
0716 REM ********************************************************
0717 REM
0720 LET E$=STR$(E)
0740 GOSUB 1750
0785  C1=C1+1
0790 GOTO 440
0800 REM
0801 REM ********************************************************
0802 REM * WITH X,Y,Z ENTROPY CALCULATED, NOW CALCULATE TOTAL,
0803 REM * COMBINED ENTROPY BY ADDING X,Y,Z BIN COUNTS & DIVIDING
0804 REM * BY TOTAL SAMPLE COUNT
0805 REM ******************************************************
0806 REM
0807  E5=0
0810 FOR I=1 TO 255
0820  T(I)=T(I)/S2
0822 IF T(I)=0 THEN GO TO 840
```

```
0830  E5=E5+(T(I)*(-(LOG(T(I))/.693147)))
0840 NEXT I
0850  T9=T9/S2
0852 IF T9=0 THEN GO TO 870
0860  E5=E5+(T9*(-(LOG(T9)/.693147)))
0870  C3=8/E5
0872 REM
0873 REM *****************************************************
0874 REM * PRINT COMBINED ENTROPY TO TERM & STR RESULT IN MEM BUFFER
0875 REM *****************************************************
0876 REM
0880 PRINT
0890 PRINT "MAX COMPRESSION RATIO FOR 3 LEAD EKG SYSTEM ";C3;" : 1"
0910 PRINT
0920 PRINT "3 LEAD EKG SOURCE ENTROPY = ";E5;" BITS."
0930 PRINT
0940 LET E$=STR$(E5)
0950 GOSUB 1750
1000 LET E$=STR$(C3)
1005 GOSUB 1750
1010 REM
1020 REM *********************************************
1030 REM * NOW GET OTHER STATISTICAL VARIABLES &
1040 REM * CALCULATE CHANNEL MAX,MINS,COMPRESSION RATIO
1050 REM * OBTAINED, AND COMPRESSION TIME EFFICIENCY
1060 REM *********************************************
1070 REM
1080 REM *********************************************
1082 REM *  START WITH ACHIEVED COMPRESSION RATIO. THEN
1090 REM *  PRINT  THE TIME COMPRESSION EFFICIENCY. THIS IS
1091 REM *  THE PERCENTAGE OF THE TIME AVALIABLE THAT WAS USED
1092 REM *  TO COLLECT,COMPRESS, & CALCULATE STAT VARIABLES.
1100 REM *********************************************
1110 REM
1120 DIGITS= 4
1130  L7=6
1140  M4=PEEK(13483)+PEEK(13482)*256+PEEK(13481)*256*256
1150  D4=PEEK(13487)+PEEK(13486)*256+PEEK(13485)*256*256
1160  D4=PEEK(13484)*256*256*256+D4
1170  C4=D4/M4
1180 LET E$=STR$(C4)
1190 GOSUB 1750
1200 PRINT
1201 PRINT "COMPRESSION RATIO ACHIEVED = ";C4;" : 1"
1210 IF G$="S" THEN GOTO 1260
1211 REM
1212 REM ************************************************
1213 REM * CALCULATE OVERALL COMPRESSION EFFICIENCY
1214 REM ************************************************
1215 REM
1216 DIGITS= 1
1217  C6=((C4-1)/(C3-1))*100
1218 PRINT
1219 PRINT "ACHIEVED COMPRESSION EFFICIENCY = ";C6;" %"
```

```
1220 LET E$=STR$(C6)
1221  L7=4
1222 GOSUB 1750
1229 REM
1230 REM **********************************************
1240 REM *  NOW CALCULATE COMPRESSION TIME EFFECIENCY
1250 REM *******************************************
1252 REM
1260  S4=PEEK(13461)+PEEK(13460)*256
1262  L4=PEEK(13462)
1264  L5=PEEK(13459)+PEEK(13458)*256+PEEK(13457)*256*256
1266  L5=PEEK(13456)*256*256*256+L5
1270  T4=(1-(L5/(L4*S4)))*100
1272 DIGITS= 1
1274  L7=4
1276 LET E$=STR$(T4)
1278 GOSUB 1750
1280 PRINT
1282 PRINT "COLLECTION TIME EFFICIENCY = ";T4;" %"
1284 PRINT
1290 REM
1291 REM ***********************************************
1292 REM * CALCULATE COLLECTION DURATION
1293 REM ************************************************
1294 REM
1295  T6=S4/S8
1296 PRINT
1297 PRINT "COLLECTION DURATION = ";T6;" SECONDS"
1298 PRINT "AT A SAMPLE RATE OF ";S8;" HZ"
1299 LET E$=STR$(T6)
1300 GOSUB 1750
1301 REM
1302 REM *********************************************
1310 REM *  RETRIEVE DATA MAX & MINS & CALC VOLTS
1320 REM ***********************************************
1330 REM
1339 DIGITS= 5
1340 FOR I=6 TO 1 STEP -1
1350  M9(I)=PEEK(13460+I*3)
1360 IF M9(I) > 127 THEN GOTO 1390
1370  M9(I)=M9(I)*.0390625
1380 GOTO 1400
1390  M9(I)=-(10-M9(I)*.0390625)
1400 NEXT I
1401  E1=13408
1402 FOR I4=5 TO 1 STEP -2
1403 LET E$=STR$(M9(I4))
1404  L7=7
1405 GOSUB 1750
1406  I3=I4+1
1407 LET E$=STR$(M9(I3))
1408 GOSUB 1750
1409 NEXT I4
1410 REM
```

```
1420 REM **************************************************
1430 REM *  NOW PRINT MAX & MINS TO TERM (PRINTER)
1440 REM **************************************************
1450 REM
1460 PRINT
1470 FOR I=1 TO 6 STEP 2
1480 PRINT
1490 PRINT "CHANNEL ";T$((I+1)/2);" MINIMUM = ";M9(7-I);" VOLTS."
1500 PRINT
1510 PRINT "CHANNEL ";T$((I+1)/2);" MAXIMUM = ";M9(6-I);" VOLTS."
1520 PRINT
1530 NEXT I
1540 REM
1550 REM ************************************************
1560 REM *  FINISH WITH A PRINT OF NUM OF BITS USED TO
1570 REM *  STORE X,Y,Z IN MEM WITH THIS COMPRESSION TYPE
1580 REM ************************************************
1590 REM
1600  X3=PEEK(13490)+PEEK(13489)*256+PEEK(13488)*256*256
1610  Y3=PEEK(13493)+PEEK(13492)*256+PEEK(13491)*256*256
1620  Z3=PEEK(13496)+PEEK(13495)*256+PEEK(13494)*256*256
1630  T3=PEEK(13499)+PEEK(13498)*256+PEEK(13497)*256*256
1632 DIGITS= 0
1640 PRINT
1650 PRINT "NUMBER OF BITS USED TO STORE CHANNEL X =";X3
1652 PRINT
1660 PRINT "NUMBER OF BITS USED TO STORE CHANNEL Y =";Y3
1662 PRINT
1670 PRINT "NUMBER OF BITS USED TO STORE CHANNEL Z =";Z3
1672 PRINT
1680 PRINT "NUMBER OF BITS USED TO STORE TIME =";T3
1682 PRINT
1683 PRINT
1690 GOTO 2000
1691 REM
1692 REM **************************************************
1693 REM * SUB TO PRINT OUT COMPRESSION TYPE
1694 REM **************************************************
1695 REM
1696  H=PEEK(13312)*256+PEEK(13313)
1697 IF H=20035 THEN H$(1)="NOT COMP"
1698 IF H=20035 THEN H$(2)="RESSED  "
1699 IF H=21569 THEN H$(1)="TOLAN A "
1700 IF H=21569 THEN H$(2)="        "
1701 IF H=21570 THEN H$(1)="TOLAN B "
1702 IF H=21570 THEN H$(2)="        "
1703 IF H=17487 THEN H$(1)="DOWER   "
1704 IF H=17487 THEN H$(2)="        "
1705 IF H=21584 THEN H$(1)="TURNING "
1706 IF H=21584 THEN H$(2)="POINT   "
1707 IF H=18766 THEN H$(1)="2ND ORD "
1708 IF H=18766 THEN H$(2)="INTERPOL"
1709 PRINT "COMPRESSION USED......";H$(1);H$(2)
1710 PRINT
```

```
1711 RETURN
1720 REM
1730 REM ********************************************
1732 REM * SUBROUTINE TO WRITE ASCII DATA IN E$ TO MEM
1733 REM * LOCATION POINTED TO BY E1.  WILL WRITE 8 CHAR
1734 REM * STRINGS TO MEMORY.
1739 REM ********************************************
1740 REM
1750 FOR I=1 TO L7
1760  F$=MID$(E$,I,L7)
1770  F=ASC(F$)
1780 POKE( E1,F)
1781  E1=E1+1
1790 NEXT I
1800 POKE( E1,4)
1801  E1=E1+1
1810 RETURN
1820 REM
1830 REM ******************************************
1840 REM * SUBROUTINE TO PROMPT COMMAND FOR SHORT RUN OR LONG
1850 REM * RUN. SHORT DOES NOT CALCULATE ENTROPY OR COMPRESSION
1860 REM * RATIOS.
1870 REM ****************************************************
1880 REM
1900 PRINT
1910 PRINT "ENTER S FOR SHORT RUN (NO ENTROPY CALCULATED)"
1920 PRINT "ENTER L FOR LONG RUN (WITH ENTROPY CALCULATED)"
1930 INPUT G$
1932 PRINT "ENTER COLLECTION SAMPLE RATE (IE. 500)"
1934 INPUT O$
1935  S8=VAL(O$)
1940 RETURN
1950 REM
1960 REM **********************************************
1970 REM *  END OF ENTROPY CALCULATION PROGRAM
1980 REM ***********************************************
1990 REM
2000 PRINT
2010 PRINT
2020 PRINT "ENTROPY CALCULATION & STATISTICS PRINTOUT COMPLETE"
2030 END
```

Appendix D

This appendix contains a listing of the data compressed by the TOLAN-A8 module for the thesis experiment.

```
EKG SAMPLE COLLECTION STATISTICS : PAGE 1
FILENAME. . . . . . . . . . TA1359PA
SUBJECT . . . . . . . . . . PARNELL
SAMPLING RATE . . . . . . . 500 HZ
DATE OF COLLECTION. . . . . 23 OCT 80
TIME OF COLLECTION. . . . . 1359
COMPRESSION USED. . . . . . TOLAN-A
CHANNEL X ENTROPY . . . . . 2.4871   BITS
CHANNEL Y ENTROPY . . . . . 2.4751   BITS
CHANNEL Z ENTROPY . . . . . 2.4217   BITS
TOTAL SOURCE ENTROPY. . . . 2.4796   BITS
PRESS RETURN FOR PAGE 2 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS : PAGE 2
APPROX MAX COMPRESSION
RATIO POSSIBLE. . . . . . . 3.2263   : 1
COMPRESSION RATIO
ACHIEVED. . . . . . . . . . 2.2595   : 1
ACHIEVED COMPRESSION
EFFICIENCY. . . . . . . . . 56.5  % OF MAXIMUM
COMPRESSION TIME
EFFICIENCY OBTAINED . . . . 77.8  % SMP INTERVAL
COLLECTION DURATION . . . . 26.2  SECONDS
CHANNEL X MAXIMUM . . . . . 2.50000  VOLTS
CHANNEL X MINIMUM . . . . . -0.1171  VOLTS
CHANNEL Y MAXIMUM . . . . . 2.50000  VOLTS
CHANNEL Y MINIMUM . . . . . -0.1171  VOLTS
CHANNEL Z MAXIMUM . . . . . 2.50000  VOLTS
CHANNEL Z MINIMUM . . . . . -0.1562  VOLTS
COMMENTS. . . . . . . . . . X,Y,Z IN COMMON LEAD 1
PRESS RETURN FOR PAGE 3 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS: PAGE 3
NUMBER OF SAMPLES TAKEN (SAMPNO)= 3336 (HEX)
MAXIMUM LOOP COUNT PER INTERRUPT (LPCAL)= 27 (HEX)
TOTAL WAITING LOOP COUNTS DURING COLLECTION (LOOPCT)= 0001B9DB (HEX)
TIME EFFICIENCY = (1-(LOOPCT%(SAMPNO*LPCAL)))*100
CHANNEL MAXIMUMS AND MINIMUMS
XMAX= 40 (HEX) AT SAMPLE NUMBER 0DA7 (HEX)
XMIN= FD (HEX) AT SAMPLE NUMBER 0171 (HEX)
YMAX= 40 (HEX) AT SAMPLE NUMBER 0DA7 (HEX)
YMIN= FD (HEX) AT SAMPLE NUMBER 0171 (HEX)
ZMAX= 40 (HEX) AT SAMPLE NUMBER 0DA7 (HEX)
ZMIN= FC (HEX) AT SAMPLE NUMBER 0171 (HEX)
COMPRESSION STATISTICS:
NUMBER OF MEMORY BITS AVAILABLE = 021FFC (HEX)
NUMBER OF BITS AVAILABLE TO VAR LEN CODER = 00040000 (HEX)
TOTAL NUMBER OF DATA BITS STORED = 0004CD10 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL X = 006652 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Y = 0066A8 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Z = 006160 (HEX)
NUMBER OF BITS USED TO STORE TIME OR OTHER PARAMETER = 010000 (HEX)
COMPRESSION RATIO = TOTAL DATA BITS STORED PER MEM BITS AVALIABLE
```

FILE TA1359PA
CHANNEL X AMPLITUDE DISTRIBUTION
DATA VALUE NUMBER OF OCCURRENCES

80.............0000 (HEX)
81.............0000 (HEX)
82.............0000 (HEX)
83.............0000 (HEX)
84.............0000 (HEX)
85.............0000 (HEX)
86.............0000 (HEX)
87.............0000 (HEX)
88.............0000 (HEX)
89.............0000 (HEX)
8A.............0000 (HEX)
8B.............0000 (HEX)
8C.............0000 (HEX)
8D.............0000 (HEX)
8E.............0000 (HEX)
8F.............0000 (HEX)
90.............0000 (HEX)
91.............0000 (HEX)
92.............0000 (HEX)
93.............0000 (HEX)
94.............0000 (HEX)
95.............0000 (HEX)
96.............0000 (HEX)
97.............0000 (HEX)
98.............0000 (HEX)
99.............0000 (HEX)
9A.............0000 (HEX)
9B.............0000 (HEX)
9C.............0000 (HEX)
9D.............0000 (HEX)
9E.............0000 (HEX)
9F.............0000 (HEX)
A0.............0000 (HEX)
A1.............0000 (HEX)
A2.............0000 (HEX)
A3.............0000 (HEX)
A4.............0000 (HEX)
A5.............0000 (HEX)
A6.............0000 (HEX)
A7.............0000 (HEX)
A8.............0000 (HEX)
A9.............0000 (HEX)
AA.............0000 (HEX)
AB.............0000 (HEX)
AC.............0000 (HEX)
AD.............0000 (HEX)
AE.............0000 (HEX)
AF.............0000 (HEX)
B0.............0000 (HEX)
B1.............0000 (HEX)
B2.............0000 (HEX)

B3............0000 (HEX)
B4............0000 (HEX)
B5............0000 (HEX)
B6............0000 (HEX)
B7............0000 (HEX)
B8............0000 (HEX)
B9............0000 (HEX)
BA............0000 (HEX)
BB............0000 (HEX)
BC............0000 (HEX)
BD............0000 (HEX)
BE............0000 (HEX)
BF............0000 (HEX)
C0............0000 (HEX)
C1............0000 (HEX)
C2............0000 (HEX)
C3............0000 (HEX)
C4............0000 (HEX)
C5............0000 (HEX)
C6............0000 (HEX)
C7............0000 (HEX)
C8............0000 (HEX)
C9............0000 (HEX)
CA............0000 (HEX)
CB............0000 (HEX)
CC............0000 (HEX)
CD............0000 (HEX)
CE............0000 (HEX)
CF............0000 (HEX)
D0............0000 (HEX)
D1............0000 (HEX)
D2............0000 (HEX)
D3............0000 (HEX)
D4............0000 (HEX)
D5............0000 (HEX)
D6............0000 (HEX)
D7............0000 (HEX)
D8............0000 (HEX)
D9............0000 (HEX)
DA............0000 (HEX)
DB............0000 (HEX)
DC............0000 (HEX)
DD............0000 (HEX)
DE............0000 (HEX)
DF............0000 (HEX)
E0............0000 (HEX)
E1............0000 (HEX)
E2............0000 (HEX)
E3............0000 (HEX)
E4............0000 (HEX)
E5............0000 (HEX)
E6............0000 (HEX)
E7............0000 (HEX)
E8............0000 (HEX)

E9.............0000 (HEX)
EA.............0000 (HEX)
EB.............0000 (HEX)
EC.............0000 (HEX)
ED.............0000 (HEX)
EE.............0000 (HEX)
EF.............0000 (HEX)
F0.............0000 (HEX)
F1.............0000 (HEX)
F2.............0001 (HEX)
F3.............0001 (HEX)
F4.............0003 (HEX)
F5.............000C (HEX)
F6.............0012 (HEX)
F7.............0016 (HEX)
F8.............000F (HEX)
F9.............000E (HEX)
FA.............000E (HEX)
FB.............000E (HEX)
FC.............001B (HEX)
FD.............004D (HEX)
FE.............0260 (HEX)
FF.............0961 (HEX)
00.............06A2 (HEX)
01.............09A5 (HEX)
02.............024A (HEX)
03.............005F (HEX)
04.............0027 (HEX)
05.............0011 (HEX)
06.............0003 (HEX)
07.............0008 (HEX)
08.............0008 (HEX)
09.............0004 (HEX)
0A.............0001 (HEX)
0B.............0001 (HEX)
0C.............0006 (HEX)
0D.............0003 (HEX)
0E.............0003 (HEX)
0F.............000A (HEX)
10.............0009 (HEX)
11.............0000 (HEX)
12.............0003 (HEX)
13.............0002 (HEX)
14.............0000 (HEX)
15.............0000 (HEX)
16.............0000 (HEX)
17.............0000 (HEX)
18.............0000 (HEX)
19.............0000 (HEX)
1A.............0000 (HEX)
1B.............0000 (HEX)
1C.............0000 (HEX)
1D.............0000 (HEX)
1E.............0000 (HEX)

1F............0000 (HEX)
20............0000 (HEX)
21............0000 (HEX)
22............0000 (HEX)
23............0000 (HEX)
24............0000 (HEX)
25............0000 (HEX)
26............0000 (HEX)
27............0000 (HEX)
28............0000 (HEX)
29............0000 (HEX)
2A............0000 (HEX)
2B............0000 (HEX)
2C............0000 (HEX)
2D............0000 (HEX)
2E............0000 (HEX)
2F............0000 (HEX)
30............0000 (HEX)
31............0000 (HEX)
32............0000 (HEX)
33............0000 (HEX)
34............0000 (HEX)
35............0000 (HEX)
36............0000 (HEX)
37............0000 (HEX)
38............0000 (HEX)
39............0000 (HEX)
3A............0000 (HEX)
3B............0000 (HEX)
3C............0000 (HEX)
3D............0000 (HEX)
3E............0000 (HEX)
3F............0000 (HEX)
40............0000 (HEX)
41............0000 (HEX)
42............0000 (HEX)
43............0000 (HEX)
44............0000 (HEX)
45............0000 (HEX)
46............0000 (HEX)
47............0000 (HEX)
48............0000 (HEX)
49............0000 (HEX)
4A............0000 (HEX)
4B............0000 (HEX)
4C............0000 (HEX)
4D............0000 (HEX)
4E............0000 (HEX)
4F............0000 (HEX)
50............0000 (HEX)
51............0000 (HEX)
52............0000 (HEX)
53............0000 (HEX)
54............0000 (HEX)

55............0000 (HEX)
56............0000 (HEX)
57............0000 (HEX)
58............0000 (HEX)
59............0000 (HEX)
5A............0000 (HEX)
5B............0000 (HEX)
5C............0000 (HEX)
5D............0000 (HEX)
5E............0000 (HEX)
5F............0000 (HEX)
60............0000 (HEX)
61............0000 (HEX)
62............0000 (HEX)
63............0000 (HEX)
64............0000 (HEX)
65............0000 (HEX)
66............0000 (HEX)
67............0000 (HEX)
68............0000 (HEX)
69............0000 (HEX)
6A............0000 (HEX)
6B............0000 (HEX)
6C............0000 (HEX)
6D............0000 (HEX)
6E............0000 (HEX)
6F............0000 (HEX)
70............0000 (HEX)
71............0000 (HEX)
72............0000 (HEX)
73............0000 (HEX)
74............0000 (HEX)
75............0000 (HEX)
76............0000 (HEX)
77............0000 (HEX)
78............0000 (HEX)
79............0000 (HEX)
7A............0000 (HEX)
7B............0000 (HEX)
7C............0000 (HEX)
7D............0000 (HEX)
7E............0000 (HEX)
7F............0000 (HEX)

```
EKG SAMPLE COLLECTION STATISTICS : PAGE 1
FILENAME. . . . . . . . . . TA1413LU
SUBJECT . . . . . . . . . . LUTZ
SAMPLING RATE . . . . . . . 500 HZ
DATE OF COLLECTION. . . . . 23 OCT 80
TIME OF COLLECTION. . . . . 1413
COMPRESSION USED. . . . . . TOLAN-A
CHANNEL X ENTROPY . . . . . 3.7826   BITS
CHANNEL Y ENTROPY . . . . . 3.7950   BITS
CHANNEL Z ENTROPY . . . . . 3.8149   BITS
TOTAL SOURCE ENTROPY. . . . 3.8028   BITS
PRESS RETURN FOR PAGE 2 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS : PAGE 2
APPROX MAX COMPRESSION
RATIO POSSIBLE. . . . . . . 2.1036    : 1
COMPRESSION RATIO
ACHIEVED. . . . . . . . . . 1.2519    : 1
ACHIEVED COMPRESSION
EFFICIENCY. . . . . . . . . 22.8   % OF MAXIMUM
COMPRESSION TIME
EFFICIENCY OBTAINED . . . . 95.4   % SMP INTERVAL
COLLECTION DURATION . . . . 14.5   SECONDS
CHANNEL X MAXIMUM . . . . . 1.79687   VOLTS
CHANNEL X MINIMUM . . . . . -1.0156   VOLTS
CHANNEL Y MAXIMUM . . . . . 1.75781   VOLTS
CHANNEL Y MINIMUM . . . . . -1.0156   VOLTS
CHANNEL Z MAXIMUM . . . . . 1.75781   VOLTS
CHANNEL Z MINIMUM . . . . . -1.0156   VOLTS
COMMENTS. . . . . . . . . . X,Y,Z IN COMMON. LEAD 1
PRESS RETURN FOR PAGE 3 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS: PAGE 3
NUMBER OF SAMPLES TAKEN (SAMPNO)= 1C60 (HEX)
MAXIMUM LOOP COUNT PER INTERRUPT (LPCAL)= 27 (HEX)
TOTAL WAITING LOOP COUNTS DURING COLLECTION (LOOPCT)= 000031E3 (HEX)
TIME EFFICIENCY = (1-(LOOPCT%(SAMPNO*LPCAL)))*100
CHANNEL MAXIMUMS AND MINIMUMS
XMAX= 2E (HEX) AT SAMPLE NUMBER 0E7F (HEX)
XMIN= E6 (HEX) AT SAMPLE NUMBER 11C4 (HEX)
YMAX= 2D (HEX) AT SAMPLE NUMBER 0E7F (HEX)
YMIN= E6 (HEX) AT SAMPLE NUMBER 11C4 (HEX)
ZMAX= 2D (HEX) AT SAMPLE NUMBER 0E7F (HEX)
ZMIN= E6 (HEX) AT SAMPLE NUMBER 11C4 (HEX)
COMPRESSION STATISTICS:
NUMBER OF MEMORY BITS AVAILABLE = 021FF0 (HEX)
NUMBER OF BITS AVAILABLE TO VAR LEN CODER = 00032040 (HEX)
TOTAL NUMBER OF DATA BITS STORED = 0002A900 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL X = 007786 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Y = 0077F8 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Z = 00764C (HEX)
NUMBER OF BITS USED TO STORE TIME OR OTHER PARAMETERS = 00CA50 (HEX)
COMPRESSION RATIO = TOTAL DATA BITS STORED PER MEM BITS AVALIABLE
```

FILE TA1413LU
CHANNEL X AMPLITUDE DISTRIBUTION

DATA VALUE	NUMBER OF OCCURENCES
80	............0000 (HEX)
81	............0000 (HEX)
82	............0000 (HEX)
83	............0000 (HEX)
84	............0000 (HEX)
85	............0000 (HEX)
86	............0000 (HEX)
87	............0000 (HEX)
88	............0000 (HEX)
89	............0000 (HEX)
8A	............0000 (HEX)
8B	............0000 (HEX)
8C	............0000 (HEX)
8D	............0000 (HEX)
8E	............0000 (HEX)
8F	............0000 (HEX)
90	............0000 (HEX)
91	............0000 (HEX)
92	............0000 (HEX)
93	............0000 (HEX)
94	............0000 (HEX)
95	............0000 (HEX)
96	............0000 (HEX)
97	............0000 (HEX)
98	............0000 (HEX)
99	............0000 (HEX)
9A	............0000 (HEX)
9B	............0000 (HEX)
9C	............0000 (HEX)
9D	............0000 (HEX)
9E	............0000 (HEX)
9F	............0000 (HEX)
A0	............0000 (HEX)
A1	............0000 (HEX)
A2	............0000 (HEX)
A3	............0000 (HEX)
A4	............0000 (HEX)
A5	............0000 (HEX)
A6	............0000 (HEX)
A7	............0000 (HEX)
A8	............0000 (HEX)
A9	............0000 (HEX)
AA	............0000 (HEX)
AB	............0000 (HEX)
AC	............0000 (HEX)
AD	............0000 (HEX)
AE	............0000 (HEX)
AF	............0000 (HEX)
B0	............0000 (HEX)
B1	............0000 (HEX)
B2	............0000 (HEX)

B3............0000 (HEX)
B4............0000 (HEX)
B5............0000 (HEX)
B6............0000 (HEX)
B7............0000 (HEX)
B8............0000 (HEX)
B9............0000 (HEX)
BA............0000 (HEX)
BB............0000 (HEX)
BC............0000 (HEX)
BD............0000 (HEX)
BE............0000 (HEX)
BF............0000 (HEX)
C0............0000 (HEX)
C1............0000 (HEX)
C2............0000 (HEX)
C3............0000 (HEX)
C4............0000 (HEX)
C5............0000 (HEX)
C6............0000 (HEX)
C7............0000 (HEX)
C8............0000 (HEX)
C9............0000 (HEX)
CA............0000 (HEX)
CB............0000 (HEX)
CC............0000 (HEX)
CD............0000 (HEX)
CE............0000 (HEX)
CF............0000 (HEX)
D0............0000 (HEX)
D1............0000 (HEX)
D2............0000 (HEX)
D3............0000 (HEX)
D4............0000 (HEX)
D5............0000 (HEX)
D6............0000 (HEX)
D7............0000 (HEX)
D8............0000 (HEX)
D9............0000 (HEX)
DA............0000 (HEX)
DB............0000 (HEX)
DC............0000 (HEX)
DD............0000 (HEX)
DE............0000 (HEX)
DF............0000 (HEX)
E0............0000 (HEX)
E1............0000 (HEX)
E2............0000 (HEX)
E3............0000 (HEX)
E4............0000 (HEX)
E5............0000 (HEX)
E6............0000 (HEX)
E7............0000 (HEX)
E8............0000 (HEX)

E9.............0000 (HEX)
EA.............0001 (HEX)
[illegible].............0000 (HEX)
EC.............0000 (HEX)
ED.............0000 (HEX)
EE.............0000 (HEX)
EF.............0001 (HEX)
F0.............0000 (HEX)
F1.............0000 (HEX)
F2.............0001 (HEX)
F3.............0006 (HEX)
F4.............0007 (HEX)
F5.............0012 (HEX)
F6.............0012 (HEX)
F7.............0022 (HEX)
F8.............0035 (HEX)
F9.............0050 (HEX)
FA.............0082 (HEX)
FB.............00CC (HEX)
FC.............0136 (HEX)
FD.............01D9 (HEX)
FE.............02EF (HEX)
FF.............03D0 (HEX)
00.............0164 (HEX)
01.............040A (HEX)
02.............02BB (HEX)
03.............01E4 (HEX)
04.............0132 (HEX)
05.............00AD (HEX)
06.............006B (HEX)
07.............0058 (HEX)
08.............0041 (HEX)
09.............001F (HEX)
0A.............0015 (HEX)
0B.............0013 (HEX)
0C.............000C (HEX)
0D.............000A (HEX)
0E.............0002 (HEX)
0F.............0002 (HEX)
10.............0000 (HEX)
11.............0000 (HEX)
12.............0001 (HEX)
13.............0000 (HEX)
14.............0000 (HEX)
15.............0000 (HEX)
16.............0000 (HEX)
17.............0001 (HEX)
18.............0000 (HEX)
19.............0000 (HEX)
1A.............0000 (HEX)
1B.............0000 (HEX)
1C.............0000 (HEX)
1D.............0000 (HEX)
1E.............0000 (HEX)

1F............0000 (HEX)
20............0000 (HEX)
21............0000 (HEX)
22............0000 (HEX)
23............0000 (HEX)
24............0000 (HEX)
25............0000 (HEX)
26............0000 (HEX)
27............0000 (HEX)
28............0000 (HEX)
29............0000 (HEX)
2A............0000 (HEX)
2B............0000 (HEX)
2C............0000 (HEX)
2D............0000 (HEX)
2E............0000 (HEX)
2F............0000 (HEX)
30............0000 (HEX)
31............0000 (HEX)
32............0000 (HEX)
33............0000 (HEX)
34............0000 (HEX)
35............0000 (HEX)
36............0000 (HEX)
37............0000 (HEX)
38............0000 (HEX)
39............0000 (HEX)
3A............0000 (HEX)
3B............0000 (HEX)
3C............0000 (HEX)
3D............0000 (HEX)
3E............0000 (HEX)
3F............0000 (HEX)
40............0000 (HEX)
41............0000 (HEX)
42............0000 (HEX)
43............0000 (HEX)
44............0000 (HEX)
45............0000 (HEX)
46............0000 (HEX)
47............0000 (HEX)
48............0000 (HEX)
49............0000 (HEX)
4A............0000 (HEX)
4B............0000 (HEX)
4C............0000 (HEX)
4D............0000 (HEX)
4E............0000 (HEX)
4F............0000 (HEX)
50............0000 (HEX)
51............0000 (HEX)
52............0000 (HEX)
53............0000 (HEX)
54............0000 (HEX)

55............0000 (HEX)
56............0000 (HEX)
57............0000 (HEX)
58............0000 (HEX)
59............0000 (HEX)
5A............0000 (HEX)
5B............0000 (HEX)
5C............0000 (HEX)
5D............0000 (HEX)
5E............0000 (HEX)
5F............0000 (HEX)
60............0000 (HEX)
61............0000 (HEX)
62............0000 (HEX)
63............0000 (HEX)
64............0000 (HEX)
65............0000 (HEX)
66............0000 (HEX)
67............0000 (HEX)
68............0000 (HEX)
69............0000 (HEX)
6A............0000 (HEX)
6B............0000 (HEX)
6C............0000 (HEX)
6D............0000 (HEX)
6E............0000 (HEX)
6F............0000 (HEX)
70............0000 (HEX)
71............0000 (HEX)
72............0000 (HEX)
73............0000 (HEX)
74............0000 (HEX)
75............0000 (HEX)
76............0000 (HEX)
77............0000 (HEX)
78............0000 (HEX)
79............0000 (HEX)
7A............0000 (HEX)
7B............0000 (HEX)
7C............0000 (HEX)
7D............0000 (HEX)
7E............0000 (HEX)
7F............0000 (HEX)

```
EKG SAMPLE COLLECTION STATISTICS : PAGE 1
FILENAME. . . . . . . . . . TA1448L
[illegible] . . . . . . . . . [illegible]
SAMPLING RATE . . . . . . . 500 HZ
DATE OF COLLECTION. . . . . 23 OCT 80
TIME OF COLLECTION. . . . . 1449
COMPRESSION USED. . . . . . TOLAN-A
CHANNEL X ENTROPY . . . . . 2.9299   BITS
CHANNEL Y ENTROPY . . . . . 2.9243   BITS
CHANNEL Z ENTROPY . . . . . 2.9564   BITS
TOTAL SOURCE ENTROPY. . . . 2.9436   BITS
PRESS RETURN FOR PAGE 2 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS : PAGE 2
APPROX MAX COMPRESSION
RATIO POSSIBLE. . . . . . . 2.7176    : 1
COMPRESSION RATIO
ACHIEVED. . . . . . . . . . 1.6008    : 1
ACHIEVED COMPRESSION
EFFICIENCY. . . . . . . . . 34.9  % OF MAXIMUM
COMPRESSION TIME
EFFICIENCY OBTAINED . . . . 90.6  % SMP INTERVAL
COLLECTION DURATION . . . . 18.5  SECONDS
CHANNEL X MAXIMUM . . . . . 1.79687  VOLTS
CHANNEL X MINIMUM . . . . . -0.7421  VOLTS
CHANNEL Y MAXIMUM . . . . . 1.79687  VOLTS
CHANNEL Y MINIMUM . . . . . -0.7421  VOLTS
CHANNEL Z MAXIMUM . . . . . 1.79687  VOLTS
CHANNEL Z MINIMUM . . . . . -0.7812  VOLTS
COMMENTS. . . . . . . . . . X,Y,Z IN COMMON. LEAD 1
PRESS RETURN FOR PAGE 3 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS: PAGE 3
NUMBER OF SAMPLES TAKEN (SAMPNO)= 2448 (HEX)
MAXIMUM LOOP COUNT PER INTERRUPT (LPCAL)= 27 (HEX)
TOTAL WAITING LOOP COUNTS DURING COLLECTION (LOOPCT)= 0000845C (HEX)
TIME EFFICIENCY = (1-(LOOPCT%(SAMPNO*LPCAL)))*100
CHANNEL MAXIMUMS AND MINIMUMS
XMAX= 2E (HEX) AT SAMPLE NUMBER 0D0A (HEX)
XMIN= ED (HEX) AT SAMPLE NUMBER 1D0A (HEX)
YMAX= 2E (HEX) AT SAMPLE NUMBER 0D0A (HEX)
YMIN= ED (HEX) AT SAMPLE NUMBER 1D0A (HEX)
ZMAX= 2E (HEX) AT SAMPLE NUMBER 0D0A (HEX)
ZMIN= EC (HEX) AT SAMPLE NUMBER 1D0A (HEX)
COMPRESSION STATISTICS:
NUMBER OF MEMORY BITS AVAILABLE = 021FF0 (HEX)
NUMBER OF BITS AVAILABLE TO VAR LEN CODER = 0003A[illegible] (HEX)
TOTAL NUMBER OF DATA BITS STORED = 000366C0 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL X = 0069AA (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Y = 00693E (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Z = 006739 (HEX)
NUMBER OF BITS USED TO STORE TIME OR OTHER PARAMETER = 00EB98 (HEX)
COMPRESSION RATIO = TOTAL DATA BITS STORED PER MEM BITS AVAILABLE
```

FILE TA1448LT
CHANNEL X AMPLITUDE DISTRIBUTION
DATA VALUE NUMBER OF OCCURRENCES

80.............0000 (HEX)
81.............0000 (HEX)
82.............0000 (HEX)
83.............0000 (HEX)
84.............0000 (HEX)
85.............0000 (HEX)
86.............0000 (HEX)
87.............0000 (HEX)
88.............0000 (HEX)
89.............0000 (HEX)
8A.............0000 (HEX)
8B.............0000 (HEX)
8C.............0000 (HEX)
8D.............0000 (HEX)
8E.............0000 (HEX)
8F.............0000 (HEX)
90.............0000 (HEX)
91.............0000 (HEX)
92.............0000 (HEX)
93.............0000 (HEX)
94.............0000 (HEX)
95.............0000 (HEX)
96.............0000 (HEX)
97.............0000 (HEX)
98.............0000 (HEX)
99.............0000 (HEX)
9A.............0000 (HEX)
9B.............0000 (HEX)
9C.............0000 (HEX)
9D.............0000 (HEX)
9E.............0000 (HEX)
9F.............0000 (HEX)
A0.............0000 (HEX)
A1.............0000 (HEX)
A2.............0000 (HEX)
A3.............0000 (HEX)
A4.............0000 (HEX)
A5.............0000 (HEX)
A6.............0000 (HEX)
A7.............0000 (HEX)
A8.............0000 (HEX)
A9.............0000 (HEX)
AA.............0000 (HEX)
AB.............0000 (HEX)
AC.............0000 (HEX)
AD.............0000 (HEX)
AE.............0000 (HEX)
AF.............0000 (HEX)
B0.............0000 (HEX)
B1.............0000 (HEX)
B2.............0000 (HEX)

B3............0000 (HEX)
B4............0000 (HEX)
B5............0000 (HEX)
B6............0000 (HEX)
B7............0000 (HEX)
B8............0000 (HEX)
B9............0000 (HEX)
BA............0000 (HEX)
BB............0000 (HEX)
BC............0000 (HEX)
BD............0000 (HEX)
BE............0000 (HEX)
BF............0000 (HEX)
C0............0000 (HEX)
C1............0000 (HEX)
C2............0000 (HEX)
C3............0000 (HEX)
C4............0000 (HEX)
C5............0000 (HEX)
C6............0000 (HEX)
C7............0000 (HEX)
C8............0000 (HEX)
C9............0000 (HEX)
CA............0000 (HEX)
CB............0000 (HEX)
CC............0000 (HEX)
CD............0000 (HEX)
CE............0000 (HEX)
CF............0000 (HEX)
D0............0000 (HEX)
D1............0000 (HEX)
D2............0000 (HEX)
D3............0000 (HEX)
D4............0000 (HEX)
D5............0000 (HEX)
D6............0000 (HEX)
D7............0000 (HEX)
D8............0000 (HEX)
D9............0000 (HEX)
DA............0000 (HEX)
DB............0000 (HEX)
DC............0000 (HEX)
DD............0000 (HEX)
DE............0000 (HEX)
DF............0000 (HEX)
E0............0000 (HEX)
E1............0000 (HEX)
E2............0000 (HEX)
E3............0000 (HEX)
E4............0000 (HEX)
E5............0000 (HEX)
E6............0000 (HEX)
E7............0000 (HEX)
E8............0000 (HEX)

E9.............0000 (HEX)
EA.............0000 (HEX)
EB.............0000 (HEX)
EC.............0000 (HEX)
ED.............0000 (HEX)
EE.............0000 (HEX)
EF.............0000 (HEX)
F0.............0000 (HEX)
F1.............0000 (HEX)
F2.............0000 (HEX)
F3.............0000 (HEX)
F4.............0000 (HEX)
F5.............0000 (HEX)
F6.............0002 (HEX)
F7.............0001 (HEX)
F8.............0003 (HEX)
F9.............0007 (HEX)
FA.............001C (HEX)
FB.............0034 (HEX)
FC.............008F (HEX)
FD.............0180 (HEX)
FE.............03F7 (HEX)
FF.............06A2 (HEX)
00.............030D (HEX)
01.............0714 (HEX)
02.............03F0 (HEX)
03.............018A (HEX)
04.............008A (HEX)
05.............002A (HEX)
06.............0013 (HEX)
07.............0006 (HEX)
08.............0002 (HEX)
09.............0002 (HEX)
0A.............0002 (HEX)
0B.............0000 (HEX)
0C.............0000 (HEX)
0D.............0000 (HEX)
0E.............0000 (HEX)
0F.............0000 (HEX)
10.............0000 (HEX)
11.............0000 (HEX)
12.............0000 (HEX)
13.............0000 (HEX)
14.............0000 (HEX)
15.............0000 (HEX)
16.............0000 (HEX)
17.............0000 (HEX)
18.............0000 (HEX)
19.............0000 (HEX)
1A.............0000 (HEX)
1B.............0000 (HEX)
1C.............0000 (HEX)
1D.............0000 (HEX)
1E.............0000 (HEX)

1F............0000 (HEX)
20............0000 (HEX)
21............[illegible]
22............0000 (HEX)
23............0000 (HEX)
24............0000 (HEX)
25............0000 (HEX)
26............0000 (HEX)
27............0000 (HEX)
28............0000 (HEX)
29............0000 (HEX)
2A............0000 (HEX)
2B............0000 (HEX)
2C............0000 (HEX)
2D............0000 (HEX)
2E............0000 (HEX)
2F............0000 (HEX)
30............0000 (HEX)
31............0000 (HEX)
32............0000 (HEX)
33............0000 (HEX)
34............0000 (HEX)
35............0000 (HEX)
36............0000 (HEX)
37............0000 (HEX)
38............0000 (HEX)
39............0000 (HEX)
3A............0000 (HEX)
3B............0000 (HEX)
3C............0000 (HEX)
3D............0000 (HEX)
3E............0000 (HEX)
3F............0000 (HEX)
40............0000 (HEX)
41............0000 (HEX)
42............0000 (HEX)
43............0000 (HEX)
44............0000 (HEX)
45............0000 (HEX)
46............0000 (HEX)
47............0000 (HEX)
48............0000 (HEX)
49............0000 (HEX)
4A............0000 (HEX)
4B............0000 (HEX)
4C............0000 (HEX)
4D............0000 (HEX)
4E............0000 (HEX)
4F............0000 (HEX)
50............0000 (HEX)
51............0000 (HEX)
52............0000 (HEX)
53............0000 (HEX)
54............0000 (HEX)

55............0000 (HEX)
56............0000 (HEX)
57............0000 (HEX)
58............0000 (HEX)
59............0000 (HEX)
5A............0000 (HEX)
5B............0000 (HEX)
5C............0000 (HEX)
5D............0000 (HEX)
5E............0000 (HEX)
5F............0000 (HEX)
60............0000 (HEX)
61............0000 (HEX)
62............0000 (HEX)
63............0000 (HEX)
64............0000 (HEX)
65............0000 (HEX)
66............0000 (HEX)
67............0000 (HEX)
68............0000 (HEX)
69............0000 (HEX)
6A............0000 (HEX)
6B............0000 (HEX)
6C............0000 (HEX)
6D............0000 (HEX)
6E............0000 (HEX)
6F............0000 (HEX)
70............0000 (HEX)
71............0000 (HEX)
72............0000 (HEX)
73............0000 (HEX)
74............0000 (HEX)
75............0000 (HEX)
76............0000 (HEX)
77............0000 (HEX)
78............0000 (HEX)
79............0000 (HEX)
7A............0000 (HEX)
7B............0000 (HEX)
7C............0000 (HEX)
7D............0000 (HEX)
7E............0000 (HEX)
7F............0000 (HEX)

```
EKG SAMPLE COLLECTION STATISTICS : PAGE 1
FILENAME. . . . . . . . . . TA1545T
[illegible] . . . . . . . . . . [illegible]
SAMPLING RATE . . . . . . . 500 HZ
DATE OF COLLECTION. . . . . 23 OCT 80
TIME OF COLLECTION. . . . . 1546
COMPRESSION USED. . . . . . TOLAN-A
CHANNEL X ENTROPY . . . . . 3.2801   BITS
CHANNEL Y ENTROPY . . . . . 3.2716   BITS
CHANNEL Z ENTROPY . . . . . 3.3188   BITS
TOTAL SOURCE ENTROPY. . . . 3.3026   BITS
PRESS RETURN FOR PAGE 2 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS : PAGE 2
APPROX MAX COMPRESSION
RATIO POSSIBLE. . . . . . . 2.4223   : 1
COMPRESSION RATIO
ACHIEVED. . . . . . . . . . 1.3617   : 1
ACHIEVED COMPRESSION
EFFICIENCY. . . . . . . . . 25.4   % OF MAXIMUM
COMPRESSION TIME
EFFICIENCY OBTAINED . . . . 93.9   % SMP INTERVAL
COLLECTION DURATION . . . . 15.8   SECONDS
CHANNEL X MAXIMUM . . . . . 3.59375  VOLTS
CHANNEL X MINIMUM . . . . . -1.4453  VOLTS
CHANNEL Y MAXIMUM . . . . . 3.59375  VOLTS
CHANNEL Y MINIMUM . . . . . -1.4453  VOLTS
CHANNEL Z MAXIMUM . . . . . 3.55468  VOLTS
CHANNEL Z MINIMUM . . . . . -1.4453  VOLTS
COMMENTS. . . . . . . . . . X,Y,Z, IN COMMON, LEAD 1
PRESS RETURN FOR PAGE 3 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS: PAGE 3
NUMBER OF SAMPLES TAKEN (SAMPNO)= 1EDD (HEX)
MAXIMUM LOOP COUNT PER INTERRUPT (LPCAL)= 26 (HEX)
TOTAL WAITING LOOP COUNTS DURING COLLECTION (LOOPCT)= 0000470E (HEX)
TIME EFFICIENCY = (1-(LOOPCT%(SAMPNO*LPCAL)))*100
CHANNEL MAXIMUMS AND MINIMUMS
XMAX= 5C (HEX) AT SAMPLE NUMBER 03DD (HEX)
XMIN= DB (HEX) AT SAMPLE NUMBER 145D (HEX)
YMAX= 5C (HEX) AT SAMPLE NUMBER 03DD (HEX)
YMIN= DB (HEX) AT SAMPLE NUMBER 145D (HEX)
ZMAX= 5B (HEX) AT SAMPLE NUMBER 03DD (HEX)
ZMIN= DB (HEX) AT SAMPLE NUMBER 145D (HEX)
COMPRESSION STATISTICS:
NUMBER OF MEMORY BITS AVAILABLE = 021FF0 (HEX)
NUMBER OF BITS AVAILABLE TO VAR LEN CODER = 00034[illegible]0 (HEX)
TOTAL NUMBER OF DATA BITS STORED = 0002E4B8 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL X = 0071DF (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Y = 007199 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Z = 007013 (HEX)
NUMBER OF BITS USED TO STORE TIME OR OTHER PARAMETER = 00D220 (HEX)
COMPRESSION RATIO = TOTAL DATA BITS STORED PER MEM BITS AVALIABLE
```

```
EKG SAMPLE COLLECTION STATISTICS : PAGE 1
FILENAME. . . . . . . . . . TA1548T
[illegible] . . . . . . . . . . [illegible]
SAMPLING RATE . . . . . . . [illegible]
DATE OF COLLECTION. . . . 23 OCT 80
TIME OF COLLECTION. . . . 1548
COMPRESSION USED. . . . . TOLAN-A
CHANNEL X ENTROPY . . . . 3.0171   BITS
CHANNEL Y ENTROPY . . . . 3.0114   BITS
CHANNEL Z ENTROPY . . . . 3.0373   BITS
TOTAL SOURCE ENTROPY. . . 3.0355   BITS
PRESS RETURN FOR PAGE 2 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS : PAGE 2
APPROX MAX COMPRESSION
RATIO POSSIBLE. . . . . . 2.6354   : 1
COMPRESSION RATIO
ACHIEVED. . . . . . . . . 1.4998   : 1
ACHIEVED COMPRESSION
EFFICIENCY. . . . . . . . 30.5  % OF MAXIMUM
COMPRESSION TIME
EFFICIENCY OBTAINED . . . 91.4  % SMP INTERVAL
COLLECTION DURATION . . . 17.4  SECONDS
CHANNEL X MAXIMUM . . . . 2.77343  VOLTS
CHANNEL X MINIMUM . . . . -1.3671  VOLTS
CHANNEL Y MAXIMUM . . . . 2.73437  VOLTS
CHANNEL Y MINIMUM . . . . -1.3671  VOLTS
CHANNEL Z MAXIMUM . . . . 2.73437  VOLTS
CHANNEL Z MINIMUM . . . . -1.3671  VOLTS
COMMENTS. . . . . . . . . X,Y,Z IN COMMON, LEAD AVL
PRESS RETURN FOR PAGE 3 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS: PAGE 3
NUMBER OF SAMPLES TAKEN (SAMPNO)= 21FE (HEX)
MAXIMUM LOOP COUNT PER INTERRUPT (LPCAL)= 27 (HEX)
TOTAL WAITING LOOP COUNTS DURING COLLECTION (LOOPCT)= 00007134 (HEX)
TIME EFFICIENCY = (1-(LOOPCT%(SAMPNO*LPCAL)))*100
CHANNEL MAXIMUMS AND MINIMUMS
XMAX= 47 (HEX) AT SAMPLE NUMBER 0A9E (HEX)
XMIN= DD (HEX) AT SAMPLE NUMBER 1447 (HEX)
YMAX= 46 (HEX) AT SAMPLE NUMBER 095A (HEX)
YMIN= DD (HEX) AT SAMPLE NUMBER 1447 (HEX)
ZMAX= 46 (HEX) AT SAMPLE NUMBER 095A (HEX)
ZMIN= DD (HEX) AT SAMPLE NUMBER 1447 (HEX)
COMPRESSION STATISTICS:
NUMBER OF MEMORY BITS AVAILABLE = 021FF0 (HEX)
NUMBER OF BITS AVAILABLE TO VAR LEN CODER = 00036[illegible] (HEX)
TOTAL NUMBER OF DATA BITS STORED = 00032FD0 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL X = 006EE8 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Y = 006E90 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Z = 006FB8 (HEX)
NUMBER OF BITS USED TO STORE TIME OR OTHER PARAMETER = 00DB48 (HEX)
COMPRESSION RATIO = TOTAL DATA BITS STORED PER MEM BITS AVAILABLE
```

```
EKG SAMPLE COLLECTION STATISTICS : PAGE 1
FILENAME. . . . . . . . . . TA1559B
[illegible] . . . . . . . . . . [illegible]
SAMPLING RATE . . . . . . 500 HZ
DATE OF COLLECTION. . . . 23 OCT 80
TIME OF COLLECTION. . . . 1559
COMPRESSION USED. . . . . TOLAN-A
CHANNEL X ENTROPY . . . . 3.0165   BITS
CHANNEL Y ENTROPY . . . . 3.0038   BITS
CHANNEL Z ENTROPY . . . . 3.0261   BITS
TOTAL SOURCE ENTROPY. . . 3.0233   BITS
PRESS RETURN FOR PAGE 2 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS : PAGE 2
APPROX MAX COMPRESSION
RATIO POSSIBLE. . . . . . 2.6460   : 1
COMPRESSION RATIO
ACHIEVED. . . . . . . . . 1.5280   : 1
ACHIEVED COMPRESSION
EFFICIENCY. . . . . . . . 32.0  % OF MAXIMUM
COMPRESSION TIME
EFFICIENCY OBTAINED . . . 93.0  % SMP INTERVAL
COLLECTION DURATION . . . 17.7  SECONDS
CHANNEL X MAXIMUM . . . . 2.92968  VOLTS
CHANNEL X MINIMUM . . . . 0.19531  VOLTS
CHANNEL Y MAXIMUM . . . . 2.89062  VOLTS
CHANNEL Y MINIMUM . . . . 0.19531  VOLTS
CHANNEL Z MAXIMUM . . . . 2.89062  VOLTS
CHANNEL Z MINIMUM . . . . 0.15625  VOLTS
COMMENTS. . . . . . . . . X,Y,Z IN COMMON, LEAD 1
PRESS RETURN FOR PAGE 3 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS: PAGE 3
NUMBER OF SAMPLES TAKEN (SAMPNO)= 22A2 (HEX)
MAXIMUM LOOP COUNT PER INTERRUPT (LPCAL)= 26 (HEX)
TOTAL WAITING LOOP COUNTS DURING COLLECTION (LOOPCT)= 00005B1E (HEX)
TIME EFFICIENCY = (1-(LOOPCT%(SAMPNO*LPCAL)))*100
CHANNEL MAXIMUMS AND MINIMUMS
XMAX= 4B (HEX) AT SAMPLE NUMBER 1FA2 (HEX)
XMIN= 05 (HEX) AT SAMPLE NUMBER 00BD (HEX)
YMAX= 4A (HEX) AT SAMPLE NUMBER 1FA2 (HEX)
YMIN= 05 (HEX) AT SAMPLE NUMBER 005C (HEX)
ZMAX= 4A (HEX) AT SAMPLE NUMBER 1FA2 (HEX)
ZMIN= 04 (HEX) AT SAMPLE NUMBER 01D5 (HEX)
COMPRESSION STATISTICS:
NUMBER OF MEMORY BITS AVAILABLE = 021FF0 (HEX)
NUMBER OF BITS AVAILABLE TO VAR LEN CODER = 0003A780 (HEX)
TOTAL NUMBER OF DATA BITS STORED = 00033F30 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL X = 006BE5 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Y = 006AC9 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Z = 006857 (HEX)
NUMBER OF BITS USED TO STORE TIME OR OTHER PARAMETER = 00E9E0 (HEX)
COMPRESSION RATIO = TOTAL DATA BITS STORED PER MEM BITS AVAILABLE
```

```
:EKG SAMPLE COLLECTION STATISTICS : PAGE 1
FILENAME. . . . . . . . . . TA1511S
[illegible] . . . . . . . . . . [illegible]
SAMPLING RATE . . . . . . . 500 HZ
DATE OF COLLECTION. . . . 1511
TIME OF COLLECTION. . . . 1511
COMPRESSION USED. . . . . TOLAN-A
CHANNEL X ENTROPY . . . . 3.3255   BITS
CHANNEL Y ENTROPY . . . . 3.3341   BITS
CHANNEL Z ENTROPY . . . . 3.3560   BITS
TOTAL SOURCE ENTROPY. . . 3.3457   BITS
PRESS RETURN FOR PAGE 2 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS : PAGE 2
APPROX MAX COMPRESSION
RATIO POSSIBLE. . . . . . 2.3911   : 1
COMPRESSION RATIO
ACHIEVED. . . . . . . . . 1.4319   : 1
ACHIEVED COMPRESSION
EFFICIENCY. . . . . . . . 31.0  % OF MAXIMUM
COMPRESSION TIME
EFFICIENCY OBTAINED . . . 94.0  % SMP INTERVAL
COLLECTION DURATION . . . 16.6  SECONDS
CHANNEL X MAXIMUM . . . . 1.32812  VOLTS
CHANNEL X MINIMUM . . . . -2.0703  VOLTS
CHANNEL Y MAXIMUM . . . . 1.32812  VOLTS
CHANNEL Y MINIMUM . . . . -2.0703  VOLTS
CHANNEL Z MAXIMUM . . . . 1.32812  VOLTS
CHANNEL Z MINIMUM . . . . -2.0703  VOLTS
COMMENTS. . . . . . . . . X,Y,Z IN COMMON. LEAD AVL
PRESS RETURN FOR PAGE 3 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS: PAGE 3
NUMBER OF SAMPLES TAKEN (SAMPNO)= 2074 (HEX)
MAXIMUM LOOP COUNT PER INTERRUPT (LPCAL)= 27 (HEX)
TOTAL WAITING LOOP COUNTS DURING COLLECTION (LOOPCT)= 00004ADC (HEX)
TIME EFFICIENCY = (1-(LOOPCT%(SAMPNO*LPCAL)))*100
CHANNEL MAXIMUMS AND MINIMUMS
XMAX= 22 (HEX) AT SAMPLE NUMBER 1171 (HEX)
XMIN= CB (HEX) AT SAMPLE NUMBER 021D (HEX)
YMAX= 22 (HEX) AT SAMPLE NUMBER 1423 (HEX)
YMIN= CB (HEX) AT SAMPLE NUMBER 021D (HEX)
ZMAX= 22 (HEX) AT SAMPLE NUMBER 1423 (HEX)
ZMIN= CB (HEX) AT SAMPLE NUMBER 00E8 (HEX)
COMPRESSION STATISTICS:
NUMBER OF MEMORY BITS AVAILABLE = 021FF0 (HEX)
NUMBER OF BITS AVAILABLE TO VAR LEN CODER = 0003[illegible] (HEX)
TOTAL NUMBER OF DATA BITS STORED = 00030AE0 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL X = 00717A (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Y = 00713D (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Z = 006[illegible]7 (HEX)
NUMBER OF BITS USED TO STORE TIME OR OTHER PARAMET[illegible] = 00F000 (HEX)
COMPRESSION RATIO = TOTAL DATA BITS STORED PER MEM BITS AVALIABLE
```

```
EKG SAMPLE COLLECTION STATISTICS : PAGE 1
FILENAME. . . . . . . . . . TA1520B
SUBJECT . . . . . . . . . . BALSAMO
SAMPLING RATE . . . . . . . 500 HZ
DATE OF COLLECTION. . . . . 23 OCT 80
TIME OF COLLECTION. . . . . 1520
COMPRESSION USED. . . . . . TOLAN-A
CHANNEL X ENTROPY . . . . . 2.6012   BITS
CHANNEL Y ENTROPY . . . . . 2.5808   BITS
CHANNEL Z ENTROPY . . . . . 2.6229   BITS
TOTAL SOURCE ENTROPY. . . . 2.6104   BITS
PRESS RETURN FOR PAGE 2 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS : PAGE 2
APPROX MAX COMPRESSION
RATIO POSSIBLE. . . . . . . 3.0645   : 1
COMPRESSION RATIO
ACHIEVED. . . . . . . . . . 1.7247   : 1
ACHIEVED COMPRESSION
EFFICIENCY. . . . . . . . . 35.1   % OF MAXIMUM
COMPRESSION TIME
EFFICIENCY OBTAINED . . . . 90.9   % SMP INTERVAL
COLLECTION DURATION . . . . 20.0   SECONDS
CHANNEL X MAXIMUM . . . . . 2.07031   VOLTS
CHANNEL X MINIMUM . . . . . -0.5468   VOLTS
CHANNEL Y MAXIMUM . . . . . 2.07031   VOLTS
CHANNEL Y MINIMUM . . . . . -0.5468   VOLTS
CHANNEL Z MAXIMUM . . . . . 2.07031   VOLTS
CHANNEL Z MINIMUM . . . . . -0.5468   VOLTS
COMMENTS. . . . . . . . . . X,Y,Z IN COMMON. LEA D 2
PRESS RETURN FOR PAGE 3 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS: PAGE 3
NUMBER OF SAMPLES TAKEN (SAMPNO)= 2717 (HEX)
MAXIMUM LOOP COUNT PER INTERRUPT (LPCAL)= 27 (HEX)
TOTAL WAITING LOOP COUNTS DURING COLLECTION (LOOPCT)= 00008A6D (HEX)
TIME EFFICIENCY = (1-(LOOPCT/(SAMPNO*LPCAL)))*100
CHANNEL MAXIMUMS AND MINIMUMS
XMAX= 35 (HEX) AT SAMPLE NUMBER 1F9E (HEX)
XMIN= F2 (HEX) AT SAMPLE NUMBER 0A63 (HEX)
YMAX= 35 (HEX) AT SAMPLE NUMBER 1F9E (HEX)
YMIN= F2 (HEX) AT SAMPLE NUMBER 054B (HEX)
ZMAX= 35 (HEX) AT SAMPLE NUMBER 1F9E (HEX)
ZMIN= F2 (HEX) AT SAMPLE NUMBER 054B (HEX)
COMPRESSION STATISTICS:
NUMBER OF MEMORY BITS AVAILABLE = 021FF0 (HEX)
NUMBER OF BITS AVAILABLE TO VAR LEN CODER = 000[illegible]0 (HEX)
TOTAL NUMBER OF DATA BITS STORED = 000[illegible]A28 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL X = 0066BE (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Y = 006666 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Z = 0063CC (HEX)
NUMBER OF BITS USED TO STORE TIME OR OTHER [illegible] = 00F630 (HEX)
COMPRESSION RATIO = TOTAL DATA BITS STORED [illegible] BITS AVALIABLE
```

```
EKG SAMPLE COLLECTION STATISTICS : PAGE 1
FILENAME. . . . . . . . . . TA1439ST
SUBJECT . . . . . . . . . . STROUP
SAMPLING RATE . . . . . . . 500 HZ
DATE OF COLLECTION. . . . . 23 OCT 80
TIME OF COLLECTION. . . . . 1439
COMPRESSION USED. . . . . . TOLAN-A
CHANNEL X ENTROPY . . . . . 3.2669   BITS
CHANNEL Y ENTROPY . . . . . 3.2756   BITS
CHANNEL Z ENTROPY . . . . . 3.3094   BITS
TOTAL SOURCE ENTROPY. . . . 3.2912   BITS
PRESS RETURN FOR PAGE 2 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS : PAGE 2
APPROX MAX COMPRESSION
RATIO POSSIBLE. . . . . . . 2.4307    : 1
COMPRESSION RATIO
ACHIEVED. . . . . . . . . . 1.4314    : 1
ACHIEVED COMPRESSION
EFFICIENCY. . . . . . . . . 30.1  % OF MAXIMUM
COMPRESSION TIME
EFFICIENCY OBTAINED . . . . 93.4  % SMP INTERVAL
COLLECTION DURATION . . . . 16.6  SECONDS
CHANNEL X MAXIMUM . . . . . 3.35937  VOLTS
CHANNEL X MINIMUM . . . . . -0.5468  VOLTS
CHANNEL Y MAXIMUM . . . . . 3.35937  VOLTS
CHANNEL Y MINIMUM . . . . . -0.5468  VOLTS
CHANNEL Z MAXIMUM . . . . . 3.35937  VOLTS
CHANNEL Z MINIMUM . . . . . -0.5859  VOLTS
COMMENTS. . . . . . . . . . X,Y,Z IN COMMON. LEAD 1
PRESS RETURN FOR PAGE 3 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS: PAGE 3
NUMBER OF SAMPLES TAKEN (SAMPNO)= 2071 (HEX)
MAXIMUM LOOP COUNT PER INTERRUPT (LPCAL)= 27 (HEX)
TOTAL WAITING LOOP COUNTS DURING COLLECTION (LOOPCT)= 0000526C (HEX)
TIME EFFICIENCY = (1-(LOOPCT/(SAMPNO*LPCAL)))*100
CHANNEL MAXIMUMS AND MINIMUMS
XMAX= 56 (HEX) AT SAMPLE NUMBER 1472 (HEX)
XMIN= F2 (HEX) AT SAMPLE NUMBER 1E5A (HEX)
YMAX= 56 (HEX) AT SAMPLE NUMBER 1472 (HEX)
YMIN= F2 (HEX) AT SAMPLE NUMBER 0FD8 (HEX)
ZMAX= 56 (HEX) AT SAMPLE NUMBER 1472 (HEX)
ZMIN= F1 (HEX) AT SAMPLE NUMBER 1E5A (HEX)
COMPRESSION STATISTICS:
NUMBER OF MEMORY BITS AVAILABLE = 021FF0 (HEX)
NUMBER OF BITS AVAILABLE TO VAR LEN CODER = 00037[illegible] (HEX)
TOTAL NUMBER OF DATA BITS STORED = 00030A98 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL X = 006E88 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Y = 006EDC (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Z = 006CF0 (HEX)
NUMBER OF BITS USED TO STORE TIME OR OTHER PARAM[illegible] = 00DDC0 (HEX)
COMPRESSION RATIO = TOTAL DATA BITS STORED PER MEM [illegible]TS AVAILABLE
```

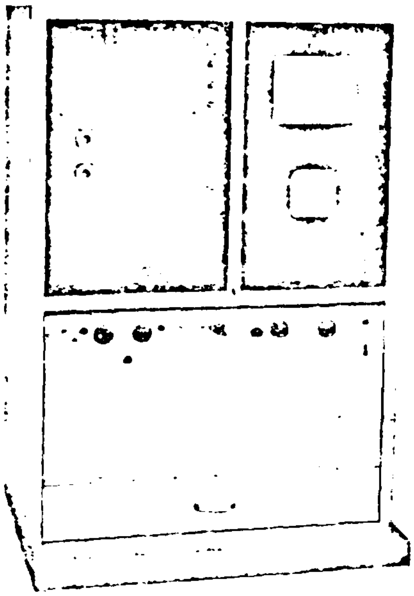
Appendix E

This appendix contains a photocopy of the specifications for the pertinent equipment used in this thesis.

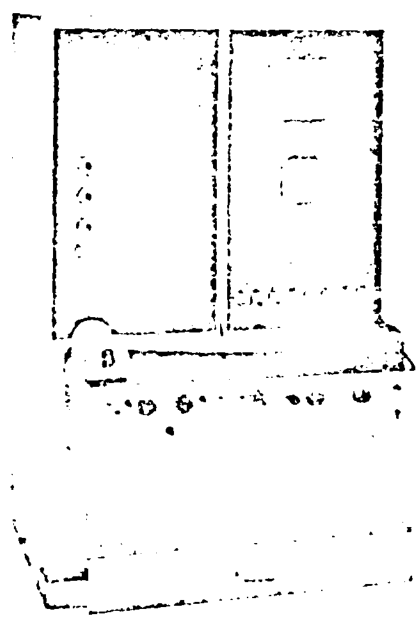
DR-12

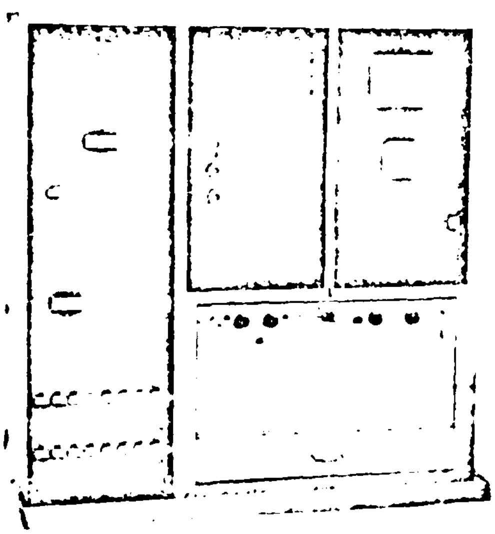
Copied from Dr. VanSlee's Diary. [illegible]
Bld 79
55C150

RESEARCH RECORDER



OPERATING INSTRUCTIONS





ELECTRONICS for MEDICINE INC

30 VIRGINIA ROAD, WHITE PLAINS, N.Y. [illegible]

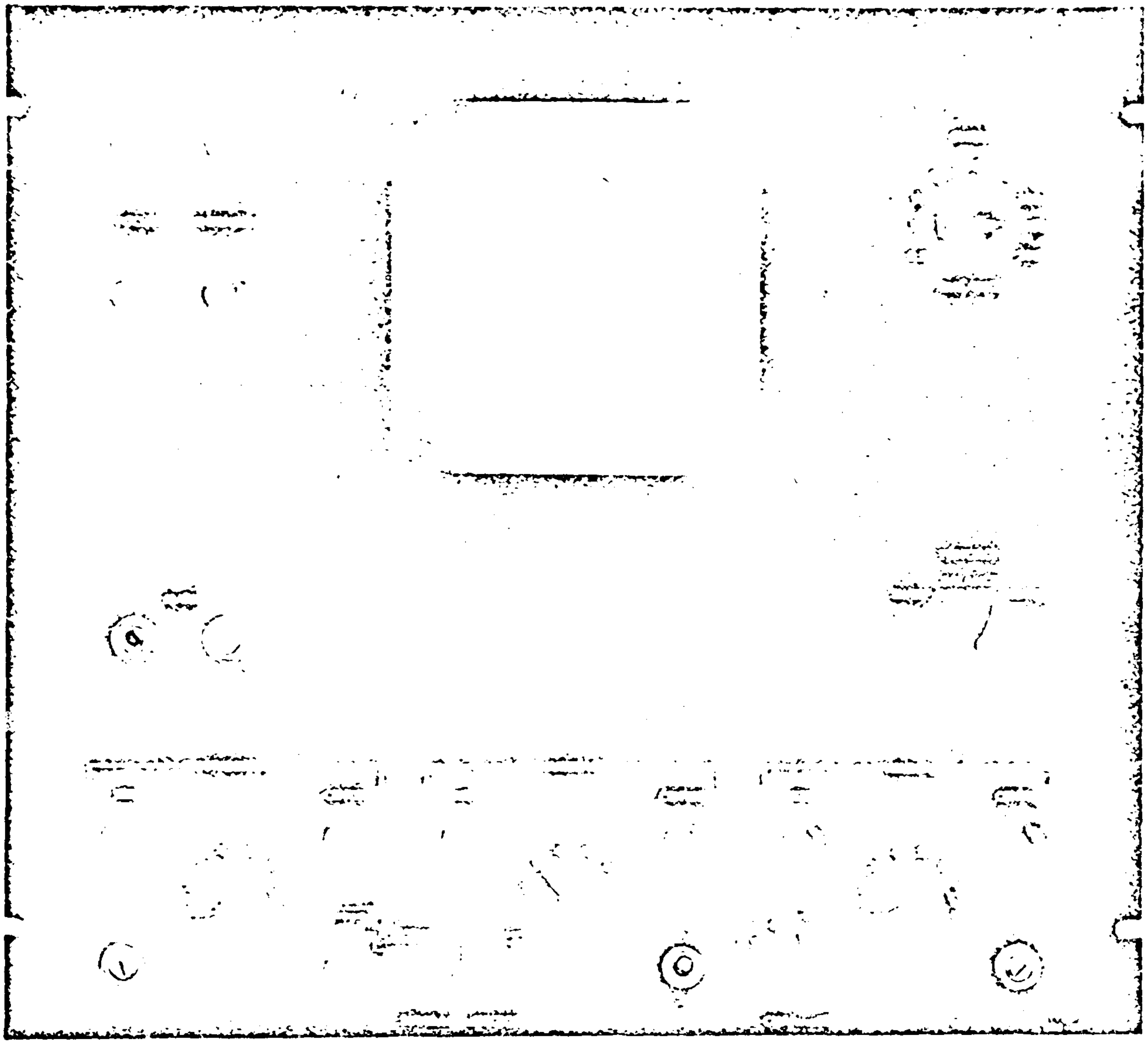
DR-8 RESEARCH RECORDER
MONITOR

The Model DR-8 Research Recorder is designed to operate from a 100-130 volt, 60 Hz line. Power consumption is about 700 watts. The circuit is protected by a 10 ampere circuit breaker which is incorporated in the power ON-OFF switch on the camera panel.

It is essential for safety and to minimize 60 Hz interference in the tracings that the Research Recorder be connected to a good "ground". A third wire is brought out at the power plug for this purpose. A good "ground" can usually be obtained at a cold water pipe.

The instrument combines as many as eight amplifiers with a cathode ray tube recording camera and two cathode ray tube monitors. The multitrace monitor displays any number of traces up to eight. The single monitor displays any one of the traces enlarged or reduced in size. It is also used for the presentation of loops, such as the vectorcardiogram or the lung compliance and lung resistance loops, and for balancing strain gauge amplifiers. All phenomena can be monitored continuously. Recording is done at will, merely by turning a switch. The display on the cathode ray tube in the camera can be identical to that on either the multitrace tube or the single trace tube. It is therefore possible to photograph either scalar tracings or loops. In actual operation, size adjustments and preliminary measurements are made by observing the monitor screens without recording. Thus, the camera runs for only short intervals.

The multitrace tube has only one beam, but because the amplifiers are connected to it through an electronic switch operating at a rate of [illegible] Hz/sec., it is possible to display as many as eight traces simultaneously. The beam switches rapidly from one trace to the next, with blanking of the cathode ray tube taking place during the switching period. Thus, each beam is on screen only 1/8th of the time, although giving the impression of being continuous. A switch is provided to make possible more frequent sampling for high speed phenomena, by reducing the number of channels displayed.



POWER switch, located on the right side of camera panel, turns the Research Recorder on. A minute or two is required before the traces on the monitor screens are stabilized. Adjustments may now be made.

INTENSITY and FOCUS control adjust the cathode ray tube beams. The upper two controls are for the multitrace tube, the lower two for the single trace tube. The intensity is set according to the brightness of the room in which the instrument is being used. The traces will be sharper at lower intensities. Focus is adjusted for the sharpest spot and should be done when the beams are approximately 1/3rd of the distance from either side of the tube to allow best overall focusing.

FUNCTION selector switch determines the presentation on the single trace tube; the multitrace tube will display scalar traces continuously. The single trace tube will display the trace from any amplifier when the FUNCTION selector switch is in the SWEEP position. In the BALANCE position, a pattern is displayed on the single trace screen which permits rapid balancing of the pressure transducers. In LOOPS, any channel can be presented on the horizontal axis of the single trace tube, and at the same time any othe[illegible] e presented on the vertical axis. This position is used for the VCG [illegible] ops (X-Y plot).

SWEEP switch determines the speed at which the traces move across the screen. The proper setting depends on the repetition rate of the phenomena displayed. Nine monitoring speeds are available: 2.5, 5, 10, 25, 50, 75, 100, 150 and 200 mm/sec.

The MULTITRACE MONITOR permits the display of up to eight channels at one time. The positions of the individual traces on the multitrace tube are controlled by the CENTER knob on the individual channels. Each of the traces can be positioned anywhere on the screen, superimposed on other traces, or can be moved off screen in either direction if it is not being used. The amplitude of the traces on the multitrace screen will depend upon the setting of the AMPLITUDE controls on the individual amplifiers.

The CHANNELS switch determines the number of amplifiers that will be displayed. The multitrace presentation is the result of an 8 channel electron switch, whose basic frequency is 48 K Hz. In the 1-8 position, each amplifier will be sampled equally 6,000 times second, or every 166 microseconds. In the 1-5 position, channels 1, 2 and 3 are sampled at 12,000 times second or every 83 microseconds. Channels 4 and 5 are sampled at the normal 6,000 times and channels 6, 7 and 8 are not displayed at all. Placing the multitrace switch at 1-6 will sample channels 1 and 2 at the 12,000 rate and 3, 4, 5 and 6 at the normal 6,000 rate. Channels 7 and 8 will not appear on the multitrace screen. Amplifiers providing fast rise time phenomena, such as heart sounds, action and nerve potentials, should be in positions 1, 2 and 3 to take advantage of the faster sampling rate and to minimize the dotting effect.

When the SUBTRACT SELECTOR switch is in the OFF position, the built-in baseline, gradient and subtract functions are inoperative. Turning switch to BASELINE will allow built in baseline to be positioned to any point on screen with POSITION control. An event MARK button and jack (for marking from a remote position) are provided. To display either baseline or gradient 1-5, 1-6, 1-8 switch must be in 1-8, since baseline or gradient trace replace channel 8. When BASELINE or GRADIENT position is selected channel 8 amplifier will not be seen.

A PRESSURE GRADIENT can be displayed on the multitrace screen together with the two pressures. The adjustment procedure is as follows:

1) Attach pressure transducers to pressure amplifiers; allow them to warm up, balance the amplifiers. Position the traces to baseline from which measurements are to be made; equalize the amplitudes. (See SGA-SGM instructions)

2) Turn the SUBTRACT SELECTOR switch to "GRADIENT". The gradient trace will now be substituted for Channel 8 and its position controlled by the baseline POSITION control.

3) Turn HORIZONTAL SIZE and SUBTRACT SIZE to zero (counterclockwise)

4) Select the channel to be subtracted from with the Horizontal switch; the channel to be subtracted with the Subtract switch.

5) Position the gradient trace to the baseline position of the two pressure channels. There are now 3 or 4 lines superimposed i.e., 2 pressure amplifiers, the difference trace (gradient) and optionally a baseline, which can be a baseline, marker trace, or any unused pressure amplifier.

6) Turn the +/- switch to +.

7) Turn HORIZONTAL SIZE slowly clockwise; note in which direction trace moves. If it moves upwards, turn HORIZONTAL CENTER counter-clockwise to return trace to its original position. Advance HORIZONTAL SIZE fully clockwise and readjust HORIZONTAL CENTER, if necessary. When the center is optimally adjusted, turning HORIZONTAL SIZE should not affect the position of trace. Return HORIZONTAL SIZE to zero. Repeat procedure with SUBTRACT SIZE and CENTER controls, returning SIZE to zero after obtaining balance.

8) Turn CALIBRATE control on pressure amplifier which was selected by HORIZONTAL SWITCH so that its trace moves approximately full screen; adjust HORIZONTAL SIZE until gradient trace coincides with the pressure trace. Remove the CALIBRATE signal from the pressure amplifier and both lines will return to their original positions.

9) Turn the CALIBRATE on that amplifier selected by SUBTRACT switch until the pressure trace moves approximately full screen. Adjust SUBTRACT SIZE until lines are superimposed. Return CALIBRATE switch to zero and both lines will return to their original positions.

10) The "Gradient" trace will now represent the sum of the two pressures. To observe the gradient, turn the +/- switch to -. Readjust gradient POSITION, if necessary.

For the H and V sections of SUBTRACT SELECTOR switch, see loops.

<u>SINGLE TRACE MONITOR</u> presentation is determined by the FUNCTION switch. Any individual amplifier can be displayed when the FUNCTION switch is in the SWEEP position. Turn the VERTICAL selector switch to the channel number representing the desired amplifier. Make certain this amplifier is visible on multitrace screen. The position of the selected trace can be controlled by the CENTER control on the channel, or by the VERTICAL CENTER control. The amplitude on the screen is adjusted by the VERTICAL SIZE control and can be 2 1/2 times the amplitude of the corresponding trace on the multitrace tube, or it

can be reduced as low as zero amplitude. When the SIZE control is at zero (completely counterclockwise), the CENTER control will have no effect. If trace is more than 1" from mid-screen, (SIZE control fully counterclockwise) turn screwdriver adjust below vertical CENTER to return baseline to mid-screen. Turning the SIZE control clockwise may move trace off screen. Note in which direction it moves (up or down) and return trace to screen with VERTICAL CENTER control. If trace moves upward, turn CENTER counterclockwise; if trace moves downward, turn CENTER clockwise.

The BALANCE position of the FUNCTION switch will display a lissajous figure, which is used as an aid in balancing a transducer in a pressure amplifier.

Turn VERTICAL selector switch to pressure amplifier (with transducer connected) to be balanced. The horizontal deflection represents the transducer excitation voltage, the vertical represents the output voltage from the transducer through a pressure amplifier.

The vertical SIZE and CENTER controls do not have any effect in the BALANCE position. When the transducer is brought into balance, there will not be any output voltage, and the lissajous presentation will be a straight horizontal line.

In LOOPS position, the VERTICAL selector switch selects the channel to be presented on the vertical axis, and the vertical SIZE and CENTER controls are used to adjust vertical amplitude and position.

The HORIZONTAL switch selects the channel to be presented on the horizontal axis, and the horizontal SIZE and CENTER controls are used to adjust horizontal amplitude and position. When the HORIZONTAL (or VERTICAL) SIZE control is turned to minimum counterclockwise, the CENTER controls will not have any effect. A dot should appear at mid-screen. If it is more than 1" from mid-screen, adjust screwdriver controls below VERTICAL CENTER or HORIZONTAL SIZE until dot is at mid-screen.

Advance HORIZONTAL SIZE clockwise and adjust HORIZONTAL CENTER to return dot to its original position near mid-screen. Proper adjustment of center is achieved when it is possible to turn horizontal SIZE from minimum counterclockwise to maximum clockwise without moving dot when there isn't any applied signal. Do the same with the VERTICAL and SUBTRACT switches; return the subtract SIZE to minimum counterclockwise.

The H or V positions of SUBTRACT SELECTOR make it possible to add or subtract the signals from two channels on either the horizontal or vertical axis when [illegible] loops. This is especially useful when measuring Lung Compliance or Lung Resistance. For example: Let us suppose it is desired to subtract from the horizontal axis. The HORIZONTAL switch selects the amplifier to be subtracted from. Then, the SUBTRACT selector switch is used to select the channel to be subtracted. The – switch is turned to –. The SUBTRACT SELECTOR is turned from [illegible] to H (horizontal). The SUBTRACT, SIZE and CENTER controls will now also be able to control horizontal amplitude and position but in a direction opposite to that of the HORIZONTAL controls. See instructions for Lung Resistance and Lung Compliance loops.

Timing and direction marking can be introduced to the loop patterns by turning the camera selector to LOOPS and choosing the appropriate TIMING rate. The camera ON-OFF switch need not be ON to display timing on loop pattern, except when photographing loop. Camera timing should be in .004 or .02 to properly display LC/LR loops; or .004 for VCG.

PMB
9/69

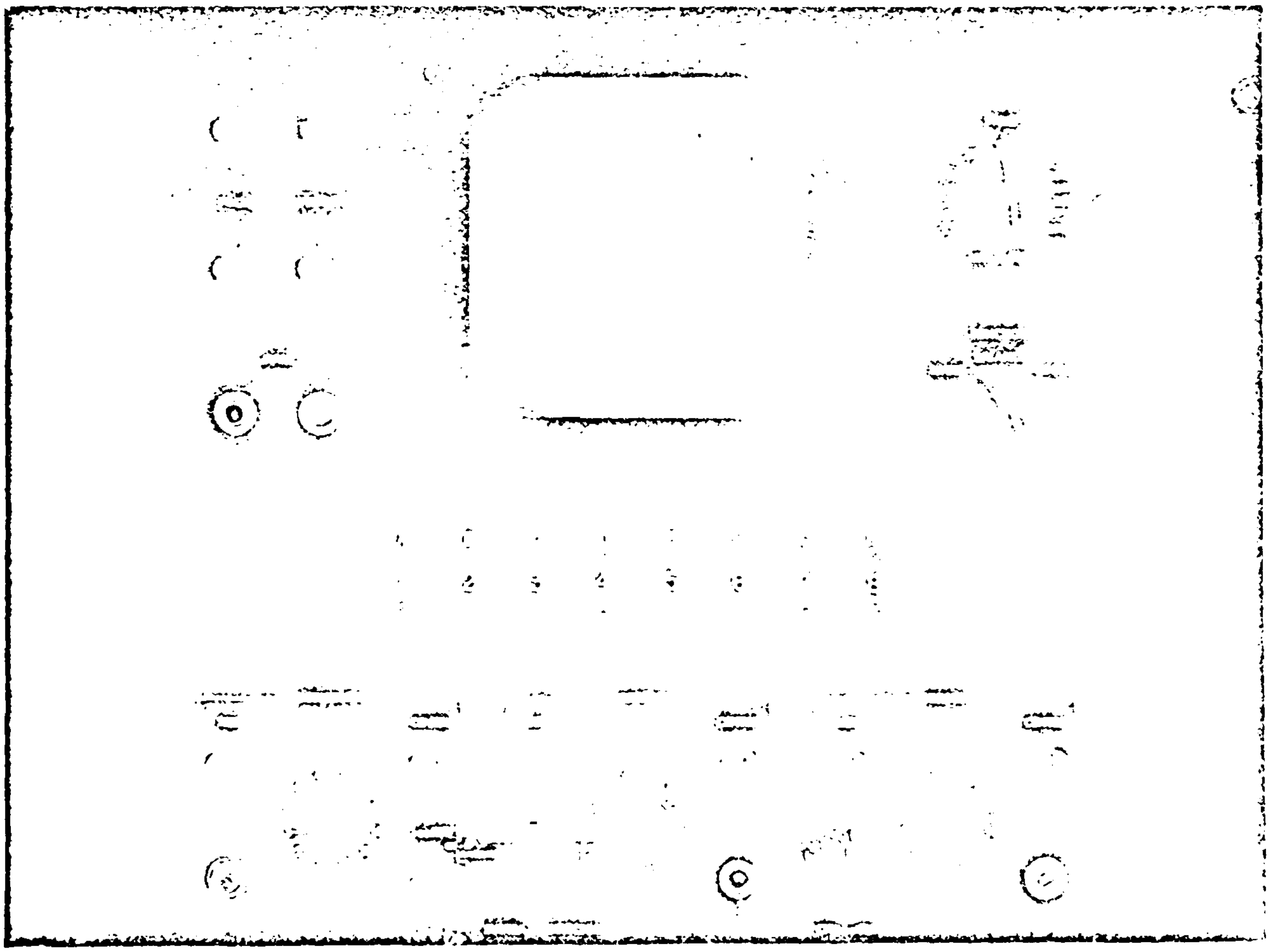
ELECTRONICS for MEDICINE, Inc., 30 Virginia Rd., White Plains, N.Y.

MONITOR

MODEL DR-12

The DR-12 recorder is a modified version of the DR-8. The DR-8 instruction plus the following information covers the operation of the DR-12.

The amplifiers in the DR-12 are numbered from 1 through 12. Ten of these, 1-10, starting from the top, are located on the left side of the recorder; numbers 11 and 12 are located beneath the monitor panel. The eight displayable channels of the recorder are lettered from A through H with a dial type switch supplied for each channel which permits any of the 12 amplifiers to be selected on any of the 8 channels. Since it is possible to select the same amplifier on more than one channel, the user should be certain that each of the dials is set to a different number or the same trace will appear on the screen more than once.



The A-E, A-F, A-H switch on the monitor panel can be used to sample two or three channels more frequently than the others. The electronic switching rate is 48,000 Hz/second; each of the 8 channels is sampled 6,000 times per second when the switch is in the A-H positior [illegible] A-E position, channels A, B, and C are sampled 12,000 times/secon [illegible] ae channels D and E are sampled 6,000 times second. Only

5 channels are operative in this position. In the A-F position, channels A and B are sampled 12,000 times/second while channels C, D, E and F are sampled 6,000 times/second. Operation is limited to 6 channels in this position. A [illegible] be sampled at these higher rates by selecting them on channels A or B.

SUBTRACT SELECTOR switch marked BASELINE, GRADIENT, OFF, H and V, when positioned on BASELINE or GRADIENT, will remove channel H (A-E, A-F, A-H switch must be on A-H) from display on multitrace screen and replace it with a baseline or gradient trace. With switch in BASELINE, the POSITION control, below the switch, can be used to set the baseline to any desired point. A MARK push button and remote MARK jack can be used as an event marker. A hand/foot switch can be patched into MARK jack, to mark events from a remote point. Placing switch on GRADIENT position will permit the display of gradients using built in gradient amplifier (POSITION control is the same as for baseline). The subtraction is done horizontally. See section on "subtraction" in DR-8 instructions. The H and V position of the SUBTRACT SELECTOR switch, permit subtraction on vector screen, either vertically or horizontally in the OFF position, subtract or GRADIENTS and BASELINE are inoperative. See DR-8 instructions.

PMB
9/69

SINETRAC SERIES

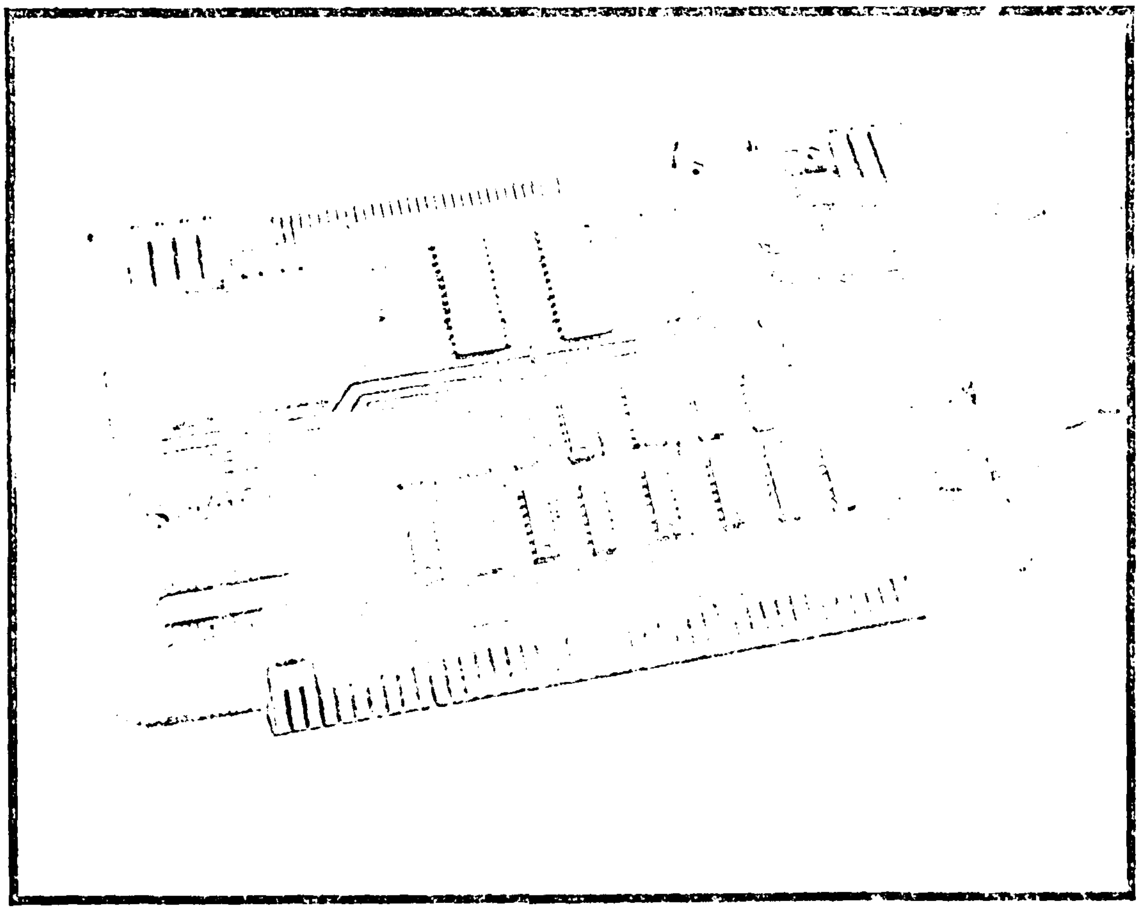
MODEL ST-6800

A/D-D/A PERIPHERAL SYSTEMS

INSTRUCTION MANUAL

Part No. 58-12140-25

JANUARY 1979



DATEL SYSTEMS, INC.
CABOT BOULEVARD, MANSFIELD, MASS 02048 TEL (617)339-9341 and 617
TWX (710)346-19[illegible] TELEX [illegible]

SPECIFICATIONS

DATA ACQUISITION SECTION

Specifications

Typical @ +25^{o}C, dynamic conditions, unless otherwise specified.

Analog Inputs

Number of Channels	32 Single-ended or 16 differential
Channel Expansion	Up to 128 single ended or 112 differential using ADC-Slave Expander Cards (Model ST-6 ADX)
Full Scale Input Ranges	0 to +5 Volts 0 to +10 Volts -5 to +5 Volts -10 to +10 Volts
Current Input Channel Range (8 Channels)	4-20 mA type
Common Mode Range	±10 Volts
Input Overvoltage	±35 Volts Max. continuous
Input Impedance	100 Megohms differential or to ground
Input Bias Current	3nA typ., 10nA max.
Input Capacitance	5pF, OFF CHANNEL, 100 pF ON CHANNEL to ground

Performance

Accuracy @ +25C	Within ±0.025% of input range
Resolution	12 Binary bits (1 part in 4096)
Nonlinearity	±1/2 LSB maximum
Differential non-linearity	±1/2 LSB maximum
Gain Error	Adjustable to zero
Offset or Zero Error	Adjustable to zero
Gain Temperature Drift (Bipolar)	Within ±10 ppm of FSR/°C
Zero Temperature Drift (Unipolar)	Within ±5 ppm of FSR/°C max.
Common Mode Rejection	70 dB min, DC to 1 kHz with 1kΩ unbalance
Power Supply Rejection	100 dB to +5V bus

Dynamic Characteristics

Typical Data Transfer	
I/O Period (Total)	36 microseconds
Throughput Period	20 microseconds
Acquisition Time	8 microseconds
A/D Conversion Time	12 microseconds
Aperture Time	100 nanoseconds
Sample/Hold Switch Feedthrough	.01% Max.
MUX Crosstalk from OFF Channels	.007% @ 1 kHz, Rs = 1K

Digital Outputs

Output Coding	Straight Binary (Unipolar) Offset Binary (Bipolar) 2's Complement (Bipolar) } Jumper Selected
Output Format	2-Byte group electrically compatible to Motorola's EXORciser bidirectional bus. Sign extension is jumper selected on bits 12 thru 15 for 2's complement units. Bits 12 thru 15 are logic zero for all other units.
Channel Addressing	Random channel addressing may be started by external interrupt input for event operation or by internal program control.
Base Address	Prewired by PC Board jumpers for one of 128 base addresses.

Data Distribution Section (D/A Analog Outputs)

Number of Channels	2 (Expandable only by stand-alone ST-6800DA Boards.)
Resolution	12 Bits
Full Scale Output	0 to +5 Volts
Voltage Ranges	0 to +10 Volts -5 to +5 Volts -10 to +10 Volts
Input Coding	Straight Binary (Unipolar) Offset Binary (Bipolar) 2's Complement (Bipolar)
Output Impedance	.05 ohm
Output Current	±5 mA min

[illegible]

Non linearity	± 1/2 LSB, maximum
Differential Nonlinearity	± 1/2 LSB, maximum
Gain Error	Adjustable to zero using pot. for each channel
Offset or Zero Error	Adjustable to zero using pot for each channel
Gain Temperature Drift	±20 ppm of output/°C
Zero Temperature Drift	(Unipolar output) ±5 ppm of FSR/°C
Offset Temperature Drift	(Bipolar output) ±10 ppm of FSR/°C
Settling Time (20V change)	4 microseconds to ± 1/2 LSB
Slew Rate	20V/usec
Power Supply Rejection	±0.02% of FSR per 1% variation

Power Consumption

1.2 ampstypical @ +5 vdc supplied from MPU bus connector. On-board DC to DC Convert supplies ±15 vdc to linear circuits.

Physical

Operating Temperature Range	0° to +70°C
Storage Temperature Range	-25°C to +85°C
Card Size	9.75"W x 5.75"H x .062"D

Room 243

Callen

Ext/55318

BRUSH

OPERATING INSTRUCTIONS

Brush Recorder Mark II

Model RD 2522 20

Specifications

Sensitivity	10 millivolts per chart line (mm). Full scale deflection from chart center ± 200 millivolts.
Sensitivity steps	.01, .02, .05, .1, .2, .5, 1, 2, 5 and 10 volts per chart line (mm). Maximum attenuator error 1% with balanced input.
Measurement range	
Single-ended input (5 megohm)	.010 volt to 400 volts
Balanced input (10 megohm)	.010 volt to 400 volts, side to side, allowable voltage off-ground at any attenuator step is 1000 x volts per chart line switch setting up to 500 volts off-ground.
Common mode rejection	Better than 1000 to 1, attenuator set in .01 volt per chart line position.
Zero line stability	Less than 1/4 chart line (mm) per hour. Total drift over eight hour period not more than 1/2 chart line (mm).
Noise	Not noticeable on chart with shorted input.
Frequency response	The recorded peak-to-peak amplitude of a constant voltage sine wave will be within [illegible] chart line (mm) of a nominal 10 lines from D.C. to 100 cps.
Maximum amplitude	40 lines (mm) peak-to-peak, D.C. to 40 cps. 20 lines (mm) peak-to-peak, D.C. to 70 cps. 10 lines (mm) peak-to-peak, D.C. to 100 cps.

Specifications

Pen Bias	Permits positioning of pen on chart, ± 20 chart lines (mm). Effective for either single-ended or balanced input.
Trace linearity	D. C. within 2% full chart width. A. C. within 3% full chart width, any frequency within limits of maximum amplitude for electric writing.
Trace width	0.006 with Model RA 2822-31 pen.
Writing method	Electric stylus.
Number of recording channels	2
Number of event channels	1 actuated by external switch. 1 actuated by panel switch.
Channel width	40 mm (40 divisions).
Chart Supply	150 feet
Chart speeds	1, 5, 25, and 125 mm per second
Chart speed regulation	Synchronous motor, direct drive.

Specifications

Operating Temperature range, ambient .... 0℃ to 55℃.

Power requirements .... 105-125 volts, 60 cps, 135 watts at 115 volts.

Transistors .... 1-5651 and 2-5687

Tubes 1-6BW4, 2-12B4A, 2-12AT7, 2-12AX7

Input terminals

Front (signal) .... Binding posts.

Rear (event Marker) .... Binding posts.

Supplies

DESCRIPTION	BRUSH PART NO.
Chart Paper, 2 channel	RA 2922-22
Electric Styli (4)	RA 2822-31
Gram Gage Assembly	227416-910
Pen Mounting Tool	126717

CHAPTER 1 GENERAL DESCRIPTION

1-1 INTRODUCTION

The M6800 EXORciser (M68SDT) is a system development tool used in the design and development of M6800 Microcomputer Systems. The EXORciser Debug and the user's system are built around the M6800 Microcomputer Family of Parts. The M6800 Microcomputer Family of Parts are discussed in the following documents.

- M6800 Microprocessor Programming Manual
- M6800 Microprocessor Applications Manual
- M6800 Microprocessor Family of Parts Data Sheets

The EXORciser may be configured in a variety of applications and with various EXORciser options. This manual, rather than discussing every possible configuration and option, discusses only the basic EXORciser with each option except the Wirewrap and Extender Modules discussed in a supplement to this manual. The basic EXORciser is discussed in Paragraph 1-5 and the options are identified in Paragraph 1-6.

This manual provides general information, installation instructions, applications information, operating procedures, and theory of operation, for Motorola's M6800 EXORciser. The M6800 EXORciser is illustrated in Figure 1-1.

1-2 EXORciser FEATURES

The features of the basic EXORciser include.

- Flexible, adaptable, and expandable design development tool
- Easy to use
- Provides the Microprocessing Unit capability for both the EXORciser and the user's system
- Saves system design and development time
- Decreases system design and development costs
- Evaluates and debugs the user's program
- Evaluates and debugs the user's system hardware

1-3 EXORciser SPECIFICATIONS

Table 1-1 identifies the basic EXORciser specifications.

TABLE 1-1. Basic EXORciser Specifications

CHARACTERISTICS	SPECIFICATIONS
Power Requirements	95-135/205-250 VAC 47-420 Hz, 250W
Word Size	
Data	8 bits
Address	16 bits
Instructions	8, 16, or 24 bits
Memory Size	64K bytes
Instruction Set	72 variable [illegible]
Clock cycle time	Selectable, 1µs or an external clock between [illegible]
Interrupt	Maskable [illegible]
Physical Characteristics	
Table top	
Length	19.25 in
Depth	17.50 in
Height	7.00 in
Rack Mountable	
Length	19.00 in
Depth	17.00 in
Height	7.00 in
Baud Rate (Switch Selectable)	110, 150, 300, [illegible], 1200, 2400, 4800, or 9600

1-4 FUNCTIONAL DESCRIPTION

The M6800 EXORciser, illustrated in Figure 1-2, [illegible] off the shelf systems [illegible] M6800 Micro[illegible] Family of Parts. The EXORciser may be easily tailored to meet the user's needs in the design and development of his system. Its modular design reduces the time required to develop a system and at the same time, provides great flexibility in configuring an emulation (functional representation) of the user's system. The EXORciser's EXbug Firmware, through its debug and program control features, minimizes the time required to develop user's systems. The EXbug firmware provides the EXORciser with the capability to:

- Display the contents of the MPU registers
- Step through the program
- Trace through user's programs to locate problem areas
- Stop the program on a selected program step
- Provide an oscilloscope trigger signal on a selected program step
- Abort from the user's program and return to the EXbug control program on command
- Reinitialize the EXORciser on command
- Change the contents of memory

The user communicates with the EXORciser in one of two ways

- Through a RS-232C or TTY terminal
- Through the EXORciser front panel controls and indicators

The terminal device permits the user to communicate directly with the EXORciser's EXbug Firmware. The EXORciser's front panel permits the user to apply power to the EXORciser, to abort (exit) the EXORciser from a routine, and to initialize and restart the EXORciser. The EXORciser's unique front panel was designed to incorporate future EXORciser options such as data keys and displays.

1-5 BASIC EXORciser SYSTEM (FIGURES 1-2 AND 1-3)

The basic EXORciser (M68SDT) consists of the MPU Module, the Debug Module, the Baud Rate Module, the Power Supply, and the chassis. These modules are built around the M6800 Microcomputer Family of Parts (MC6800 Microprocessing Unit, MC6820 Peripheral Interface Adapter, MC6850 Asynchronous Communications Interface Adapter, MCM6810 Random Access Memory, and MCM6830 Read Only Memory devices).

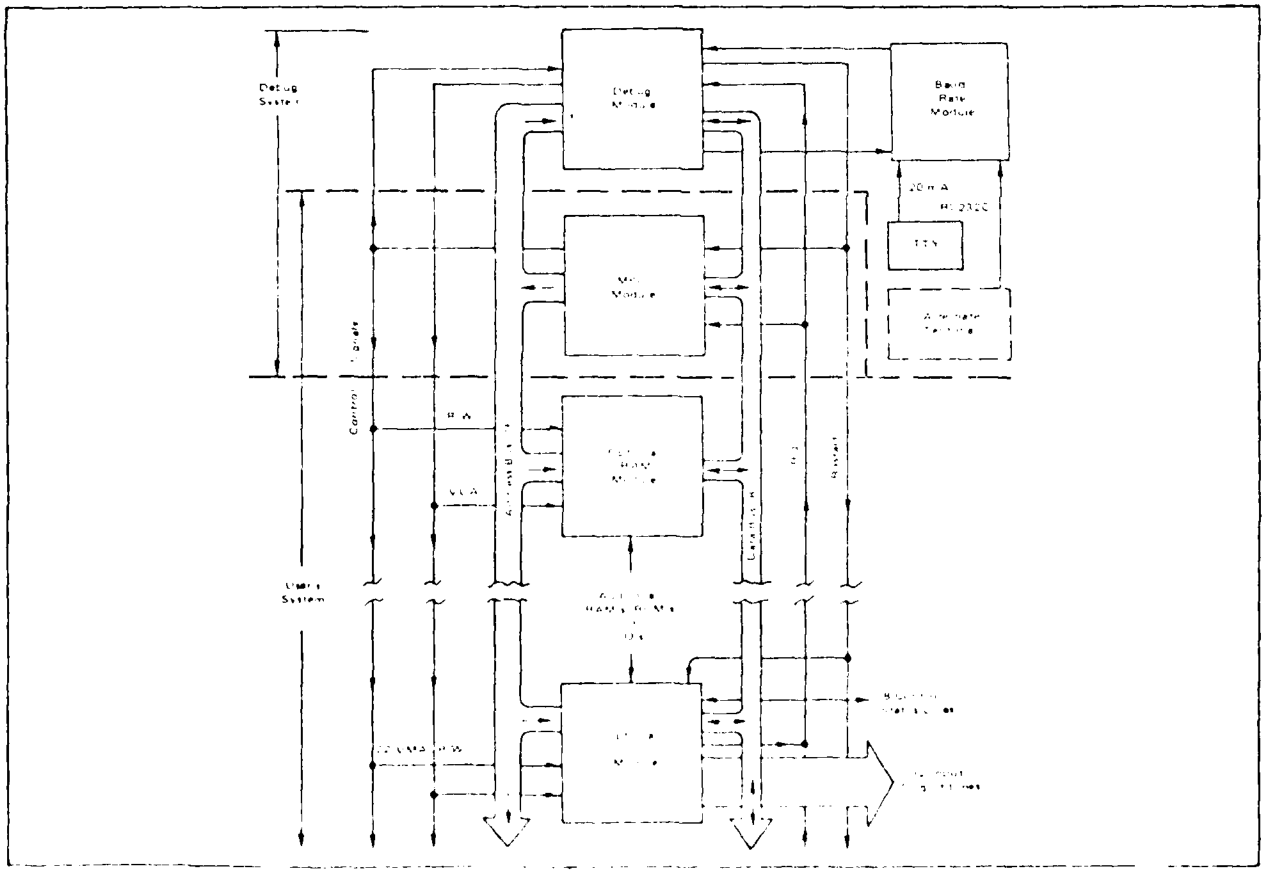


FIGURE 1-2 EXORciser Simplified Block Diagram

The MPU Module (Figure 1-4) incorporates the MC6800 Microcomputer Unit (MPU) and the system clock. This module, as illustrated in Figure 1-2, serves a dual [illegible] [illegible] tem. The MPU Module also initiates an EXORciser restart operation and initializes the EXORciser. The MC6800 Microprocessing Unit is an 8-bit parallel device capable of addressing 64k bytes of memory. In addition, the MPU addresses its input and output [illegible] memory. The MPU also provides the EXORciser with 72 variable length instructions and the [illegible] [illegible] EXORciser [illegible] (delayed); the MPU Module appears exactly like a MC6800 Microprocessing Unit with unlimited TTL bus drive capability.

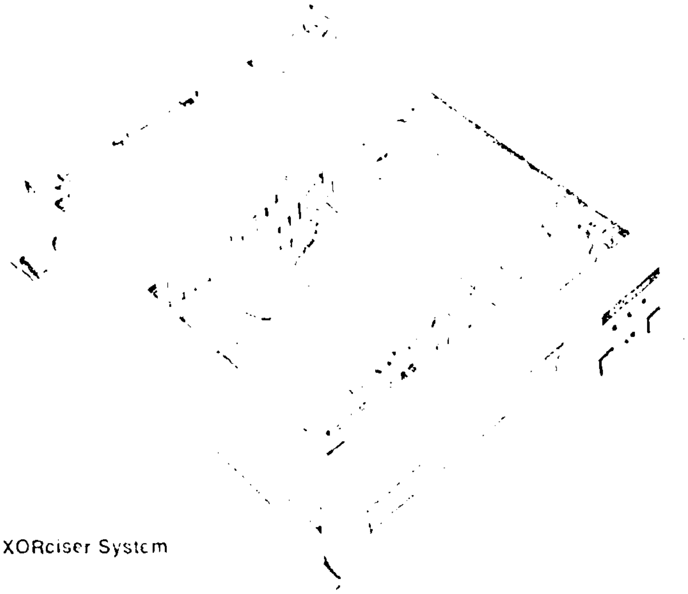


FIGURE 1-3. Basic EXORciser System

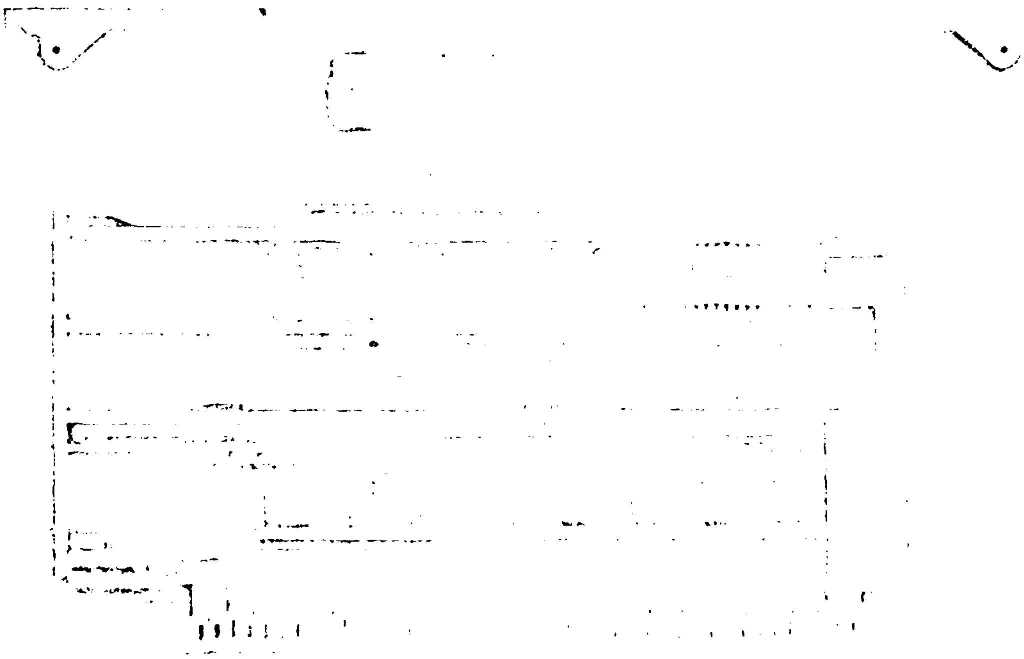


FIGURE 1-4. MPU Module

<u>Vita</u>

Melvin D. Townsend was born 5 April [illegible] in [illegible] Springs, Colorado. Upon graduation from Gen. William Mitchell High School in 1970, he enrolled in undergraduate engineering study at Colorado State University. In December, 1974, he was awarded the BSEE degree and commissioned as a second lieutenant, USAF, via AFROTC. Entering active duty immediately, he was assigned to the Foreign Technology Division (AFSC), WPAFB,Ohio. After three and one half years working technical intelligence engineering, he was transfered PCA to the AF Avionics Laboratory (AFSC) where he performed R&D engineering on electro-optical target acquisition systems. In June, 1979, he entered the Air Force Institute of Technology. Captain Townsend is a member of Eta Kappa Nu and IEEE.

Permanent Address
621 Prairie Rd.
Colorado Springs, CO
80909

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFIT/GE/EE/80D-46	2. GOVT ACCESSION NO.: AD-A100 711 / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Analysis and Performance Evaluation of Electrocardiogram Data Compression Techniques	5. TYPE OF REPORT & PERIOD COVERED: MS Thesis
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): MELVIN D. TOWNSEND, Captain USAF	8. CONTRACT OR GRANT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, Ohio 45433	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Clincal Sciences Divison USAF School of Aerospace Medicine/NGF (UAAFSAM/NGF), Brooks AFB, Texas 78235	12. REPORT DATE: December 1980
	13. NUMBER OF PAGES: 252
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17   16 JUN 1981

Fredric C. Lynch

FREDRIC C. LYNCH, Major, USAF
Director of Public Affairs

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Electrocardiogram   Data Reduction
EKG Data Compression   Source Encoding

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

EKG data compression techniques were investigated for potential real time implementation on an 8 bit Motorola 6800 microprocessor. Research indicated entropy reduction transform technique such as the Fast Fourier Transform and the discrete Karhunen-Loeve Transform were not feasible for implementation on the 6800. Two redundancy reduction (RR) techniques (TOLAN and DOMED) utilizing 2nd order difference

DD FORM 1 JAN 73 1473   EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

operations in conjunction with variable length encoding were studied in detail. One such [illegible] technique ([illegible]) [illegible] [illegible] and [illegible] in vivo [illegible] data. Analysis revealed compression ratios ranging from 1.25:1 to 2.26:1. Investigation of the poor performance of the compression algorithm showed significant degradation of the 2nd order difference "decorrelator" due to a noisy collection environment. It was concluded that real time EKG data compression is feasible on the 6800 but that time compression techniques which store a zero value sequence counter versus the value of zero are not efficient in a high noise environment.

DATE

ILMED

-8